香港中文大學
The Chinese University of Hong Kong

Final Year Project

Large Language Models for Software Systems: Directions, Opportunities, and Challenges

Author:

Chang CHEN

Houtian ZHU

Supervisor:

Prof. Michael R. LYU

LYU2407

Department of Computer Science and Engineering

Faculty of Engineering

April 23, 2025

Acknowledgement

The work described in this report was done under the supervision of Prof. Michael Lyu and Shuqing Li.

Abstract

In the field of modern software engineering, many tasks now involve inputs that go beyond plain code text, incorporating multiple modalities such as images, audio, and video. This shift introduces significant challenges in handling these diverse inputs. The emergence of large multimodal models (LMMs) offers a promising solution to this issue. However, as an emerging technology, systematic research on multimodal large models within the software engineering domain remains scarce. There is still a lack of clarity regarding the specific tasks LMMs can accomplish and their performance across these tasks.

This report conducts a comprehensive and systematic survey, categorizing and summarizing all multimodal-related problems in software engineering over the past five years, and finally constructs a complete task tree. Subsequently, we develop a modular testing framework capable of automatically measuring LMM performance based on configuration files. Within the scope of input modalities currently supported by LMMs, we select several representative tasks and evaluate their capabilities.

Our findings reveal that LMMs demonstrate surprisingly strong performance in the field of software engineering. In certain tasks, they are capable of achieving results comparable to specialized models fine-tuned for specific tasks, even without any additional fine-tuning. This highlights their significant potential for development and application in the software engineering domain.

Contents

1 Introduction

1.1 Introduction

The integration of large multimodal models (LMMs) into software systems research represents a novel frontier in artificial intelligence, blending the linguistic proficiency of large language models (LLMs) with sophisticated vision models to process and generate multimodal content. This synthesis allows LMMs to handle diverse input modalities—such as text, images, and potentially audio and video—and to produce outputs that bridge these modalities. Recent advancements in LMMs have demonstrated their capacity to perform complex reasoning tasks, often achieving strong results even in 1-shot or 0-shot scenarios. Despite these promising developments, their application within the realm of software systems remains limited, with current uses largely focused on areas like text enhancement, artwork creation, and basic summarization.

The software systems domain, encompassing software engineering, systems security, human-computer interaction (HCI), artificial intelligence, and computer graphics, involves intricate tasks that often require multimodal understanding and processing. From code analysis and software testing to user experience evaluation and cybersecurity threat detection, many of these tasks could potentially benefit from the advanced capabilities of LMMs. However, the specific challenges and opportunities presented by LMMs in these contexts have yet to be fully explored. To address this gap, a comprehensive study is necessary to understand how LMMs can enhance tasks within software systems, and to identify the architectural hurdles that must be overcome for their effective deployment.

In this vision paper, we explore the potential of LMMs to transform software systems research. We conduct a thorough review of literature spanning the past decade across related fields, constructing a task taxonomy that categorizes tasks likely to benefit from LMM capabilities. From this taxonomy, we identify representative tasks and evaluate their feasibility using a range of LMMs, such as GPT-4 Vision and Gemini Vision. Through systematic experimentation with prompt engineering, we assess the performance of these models and investigate the underlying challenges that limit their efficacy. Our work contributes to the field in the following ways:

- Task Taxonomy for LMMs in Software Systems: We develop a comprehensive taxonomy that identifies and categorizes tasks across software engineering, system security, HCI, and related fields that stand to benefit from LMM integration. This taxonomy offers a roadmap for researchers and practitioners seeking to leverage LMMs in their respective domains.
- Evaluation Framework for LMMs on Software Systems Tasks: We propose a set of evaluation criteria and experimental methods tailored to assess LMM performance on software systems tasks. By selecting representative tasks, we provide a framework for systematically test-

ing the capabilities of LMMs in real-world scenarios, including code analysis, software testing, and user experience assessment.

- Cross-Model Performance Analysis with Prompt Engineering: Using a range of LMMs, we perform a comparative analysis to understand how different models tackle similar tasks and the effectiveness of prompt engineering in enhancing their performance. This analysis sheds light on the strengths and limitations of current models, providing insights into how prompt engineering can be optimized for diverse tasks.

- Opportunities and Challenges of LMMs for Software Systems: Based on our empirical findings, we discuss the unique opportunities LMMs offer for advancing software systems research, particularly in multimodal environments such as Extended Reality (XR). We also identify the architectural challenges that hinder LMM performance, including issues related to multimodal data fusion, interpretability, and resource constraints, and propose directions for future research.

Our findings highlight the transformative potential of LMMs in software systems, paving the way for innovative applications and inspiring further exploration into this emerging field. By outlining both the current capabilities and limitations of LMMs, we aim to provide a foundation for future work that will drive the development of more intelligent, adaptive, and multimodal software solutions.

1.2 Background

Large Multimodal Models (LMMs) represent an evolution beyond traditional text-based Large Language Models (LLMs). In addition to supporting text input and output, LMMs can process inputs from multiple modalities such as images, audio, and video, generating corresponding multimodal outputs. Leading models, like GPT-4o[73], already support inputs from audio, images, and multiple image sources, while models such as Gemini[91] even handle video inputs. By extending the functionality of large models to cover multiple modalities, the scope and variety of tasks they can perform have significantly increased, including video summarization, image comprehension, and speech recognition.

The software engineering field encompasses a wide range of tasks, with the primary objective of ensuring high-quality software development and stable operation. Over the past few decades, software has evolved beyond simple command-line interfaces to incorporate graphical user interfaces (GUIs), animations, and voiceovers, which have become standard features. Consequently, relying solely on text-based code inputs has become increasingly insufficient for addressing the diverse needs of modern software systems, prompting the rise of multimodal inputs. For example, screenshots can be used to detect GUI issues[60], and video data can be analyzed to extract user gestures[8].

However, the integration of multimodal inputs into software engineering has been relatively slow. The primary challenge lies in the complexity and diversity of multimodal data, which has made it difficult for researchers to develop a unified and generalizable approach. Recent advances in machine learning have spurred efforts to combine machine learning models with multimodal input processing, such as using computer vision for object recognition in images. A key limitation of earlier approaches is that models were often task-specific, limiting their reusability across different contexts. The advent of multimodal large models offers a potential breakthrough. Numerous studiesneed citation have demonstrated the strong generalization capabilities of large models, showing that they can maintain high accuracy even with previously unseen tasks. As such, integrating LMMs into software engineering tasks is a logical next step. However, given that LMMs are still an emerging technology, there have been few attempts to explore their application in this domain. The goal of this paper is to address this gap and evaluate the performance of LMMs in software engineering tasks.

2 Related Work

In this section, we will provide an overview of how previous works utilize multimodal capabilities for problem-solving in software engineering. Then, we will discuss how LLMs can help address challenges in the software engineering domain. Finally, we will review the existing test benchmarks and evaluation criteria for assessing LMMs.

2.1 Ultizing Multimodal Ability in Software Engineering

Integrating multimodal capabilities, such as voice, gesture, and sentiment analysis, has emerged as a promising approach to enhancing software development processes and user experiences. Guglielmi et al. conducted automated tests on virtual personal assistants that use voice for interaction [36] . Qi et al. summarized recent research on gesture recognition through sensors and the analysis of image information [76] . Gandhi et al. investigated previous work on sentiment analysis, a domain encompassing the three modalities of text, vision, and audio working together to produce an effect [34] . However, these studies rely on specific mini-models or other traditional data analysis methods, which only perform relatively well on particular tasks or datasets. Those specialized models may lose good performance after migrating to other datasets or task settings of the same type [94] [77] . Our work reduces the expense of training different models for a specific problem by introducing LMMs with good generalization capabilities to handle different issues simultaneously.

2.2 LMM for Software

A branch of previous work has demonstrated that integrating LLM into the production and research of soft engineering has been a scorching trend [37] , from generating [62] [90] and pre-processing [108] [107] experimental data to using LLM as an agent for automated testing [89] [53] , all of which show that LLM has a solid potential to enhance existing soft engineering processes. Moreover, Jin et al. also illustrate the contribution that LLM can make in software design, testing, and maintenance [41]. In contrast to these studies, which only focus on specific tasks in specific domains of soft engineering and lack knowledge of what valuable tasks exist now, our work presents a systematic framework that defines what tasks are available to help optimize efficiency using LLM or LMM.

2.3 LMM Benchmark & Evaluation

LMMs combine information from different modalities, including text, vision, audio, and tactile, and analyze them to solve more complex real-world problems [104] [7] [105] . As a result, testing and evaluating LMMs' performance from different perspectives become a recent research interest. For instance, Wu et al. used the visual comprehension and language processing capabilities of GPT4v to test whether today's LMMs can support practical medical applications [99] . Cao et al. constructed Spider2-V, a test benchmark for LMM's ability to automate professional data science engineering workflows [10] . Cai et al. tested and improved the problem of the robustness of LMM's output when facing different styles of pictures [9] . These benchmarks and assessments have all achieved good performance in a single domain and can point out the shortcomings of LMM in the corresponding domain. Our work can complement the testing domains, bridging the gap of needing help harmonizing testing across domains and conducting migration tests.

2.4 Reasoning Language Model

The evolution of artificial intelligence has entered a transformative phase with the emergence of Large Reasoning Models (LRMs), an advanced paradigm built upon the foundation of Large Language Models (LLMs). While traditional LLMs excel at pattern recognition and autoregressive token prediction, their capacity for complex, structured reasoning has historically been limited. Recent breakthroughs, however, have redefined the role of language models by integrating human-like reasoning mechanisms into their architecture. This shift has unlocked unprecedented potential for solving intricate problems across domains such as mathematics, logic, and decision-making, marking the dawn of a new era in AI reasoning.

At the heart of LRMs lies the concept of ”thought”[98]—a structured sequence of intermediate tokens that simulate the step-by-step reasoning processes humans employ. Unlike conventional LLMs, which generate outputs through direct token prediction, LRMs decompose reasoning tasks into multi-step trajectories. These trajectories mimic cognitive strategies such as:

- Tree search: Exploring multiple reasoning branches to identify optimal solutions.
- Reflective thinking: Iteratively revising hypotheses based on feedback or new information.
- Analogical reasoning: Drawing parallels between problems to infer solutions.

This paradigm shift transforms LLMs from passive text generators into dynamic reasoning agents capable of deliberate, self-correcting thought processes.

3 Methodology

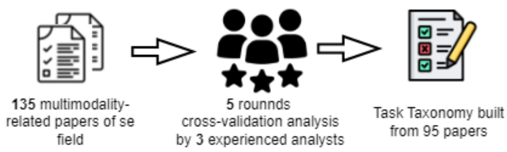
3.1 Task Taxonomy Construction

Building the prototype of taxonomy To build the prototype of the taxonomies, we have conducted systematic research on multimodality-related papers from four conferences (ICSE, FSE, ASE, and ISSTA) and two journals (TSE and TOSEM) in soft engineering over the last seven years [1]. We build a multimodality-related keyword list to screen the papers from these sources and manually collect 135 papers. To describe what types of tasks these papers in soft engineering are focusing on, we analyzed the papers according to the open coding procedures [28] used for qualitative data analysis. Specifically, we conducted a 5-round iterative manual analysis session involving three analysts with at least several years of development experience in the soft engineering field. In each iteration, every analyst separately summarizes what technical aspect of the paper belonged to the design, development, testing, maintenance, and repair [2] process of software from the Software Waterfall Model [70]. Each analyst analyzed two-thirds of the whole paper to ensure that each paper had been seen by at least two different analysts for cross-validation. Consequently, in the final iteration, we merge the research topics extracted in the previous rounds to form a prototype task tree resulting from our taxonomy. Finally, We used the results from 95 papers to build our taxonomy. At the top of our task tree are the

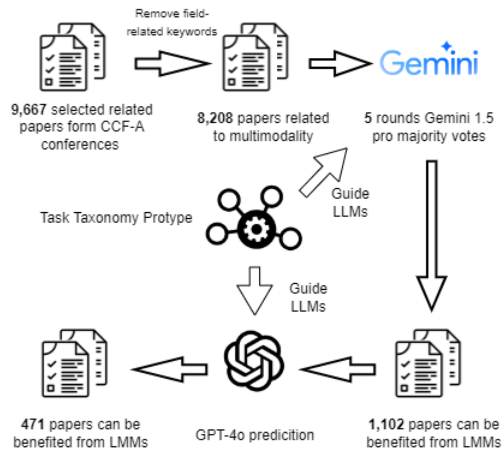
[1] from 2018.01.01 to 2024.05.15

[2] we add the "repair" process to extend our taxonomy, which initially did not exist in the Waterfall Model

five software-building processes, followed by whether they are functional or non-functional [3]. The third level of categorization is based on modal information, such as “Vision” and “Vision with Audio.” At the bottom are up to four layers of progressively more detailed descriptions of specific technical aspects.



(a) Stage 1: Building task taxonomy protype



(b) Stage 2: Guiding LLM to analyze large-scale papers and extend our task taxonomy

Figure 1: Two stages workflow of building our task taxonomy

Extending the list of papers We expand the scope of our study to encompass all 37 A-level conferences and journals as classified by the China Computer Federation[4], with the same period considered. This inclusion covers five key domains: Computer Networks, Computer Graphics and Multimedia, Artificial Intelligence, Human-Computer Interaction, and Cross-cutting/Integrated/Emerging. Subsequently, we add software-engineering-related keywords to the search keyword list to cover a broader range of papers. We also remove field-related keywords from the list for some specific domains. For example, we remove the keyword ”visual” from the list

[3] standard is followed by ISO/IEC 25002:2024
[4] https://www.ccf.org.cn/Academic_Evaluation/By_category/

of vision-related conferences, forming 8,208 pieces of paper. To reduce the number of papers and get a more concrete result, we involve the Gemini-1.5 as a judge to perform a 5-round check, where we send the paper's title and guide it to predict whether this paper may focus on multimodal tasks using following prompt3.1, and only the paper passed all the 5-round checks are selected.

prompt for analyzing software process

System : You are a computer science professor who is an expert on MLLM for SE working on survey, and you are currently working on formulating a task tree from existing papers. Analyze the problem statement and proposed solution in the paper's title and abstract to determine the following questions:

1. Which phase of the software development lifecycle it primarily addresses. Choose exactly one from:

Design (requirements analysis, UI/UX prototyping)

Development (code generation, implementation, integration)

Testing (validation, verification, quality assurance)

Maintenance (updates, optimization, documentation)

Repair (bug fixing, error recovery)

2. analyze the modalities involved in the task. i.e. a task related to GUI element testing should be classified into 'Vision with Text'. Choose the combination from: Vision/Text/Audio/Tactile (connected through 'with')

3. analyze the functionality of the problem statement in the paper. Whether it is functional (performance, accuracy) or non-functional (accessibility, security). If a task can be both considered as functional and non-functional, choose functional.

Output format: {Process: Design/Development/Testing/Maintenance/Repair, Modalities: (Vision/Text/Audio/Tactile with "with" separator), Function: Functional/Non-functional} AND DO NOT output other analysis results.

prompt for construct task tree

System : You are a helpful assistant designed to output JSON. You will be given a task tree generated from papers and a paper with its title and abstract. You are designed to answer the question:

What kind of task does the research task in the paper benefit from multi-modal AI to help process the target software/applications.

Please ensure the following rules while answering this question:

1. You have two kinds of action choices: output Matched if there is a node on the task tree matched the new task described in the paper. Otherwise output Add and the new task name in 1-5 levels to add a new node to the current task tree.
2. The first level of the task tree should use a combination (using with to connect) of terms from the modalities Vision, Text, Audio, and Tactile to describe the target modality the paper focuses on.
3. The second level of the task tree should be a broader technical concept term within its modality, avoiding the use of any specific software terms like AR, VR, or any specific software platform names (e.g., Android, Web, iOS).
4. The 'Function' in the tree describing the task should address either functional aspects, such as improvement, or non-functional aspects, such as accessibility.

prompt for construct task tree
System: Only output in 'Action': (Matched / Add), 'Function':(Functional / Non-functional) ,'1st':,'2nd':,'3rd':,'4th':,'5th:'(NA if not suitable) format. User: The Task Tree:{SubTaskTree}. Paper: {TITLE} Abstract: {ABSTRACT}

Consolidating the taxonomy This process helps to reduce the potential paper number to 1,102. Then, we perform another single-round GPT-4o prediction, where we prompt the LLM with the remaining paper's title and abstract to let the model know more about the details of the paper and make a more concrete prediction. Finally, we formulate an additional target multimodal related paper list with a size of 471, and the total paper list's size is 564.

Given that LLMs can identify latent patterns, [92] [69] we automate the expansion of our taxonomy by leveraging GPT-4o to learn these patterns from its prototypes. This process involves two distinct prompts. The first prompt instructs the LLM to analyze which software processes related to the technical aspect addressed in the paper may be relevant. In the subsequent step, we provide the subtask tree of the identified process, enabling the LLM to determine whether the aspect aligns with an existing child node or if a new child node is needed to describe it adequately. In each stage, the paper's title and abstract serve as user inputs, while the specific guidance for each part serves as system prompts. Additionally, we conduct manual checks to prune and merge misclassified results, ultimately consolidating the multimodal task taxonomy. Part of the task tree is shown in Figure. 4.

Further extending the list of papers To enhance sample robustness and mitigate potential false-negative predictions in large language model (LLM) evaluations, a revalidation process was conducted on two paper cohorts:

- 1,259 papers initially scoring 3/5 positive evaluations
- 1,177 papers initially scoring 4/5 positive evaluations

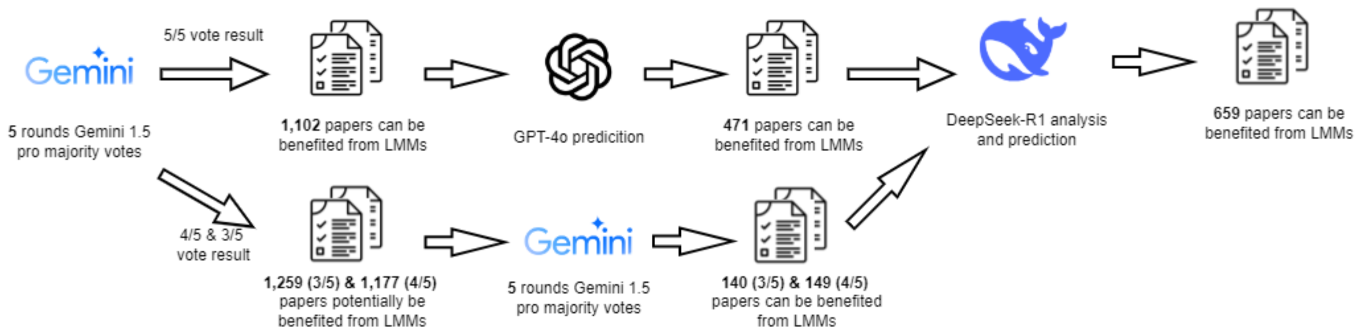


Figure 2: Updated Stage 2: multiple rounds guided LLMs prediction

These papers were further verified through five rounds of majority vote using Gemini-1.5, maintaining the predefined inclusion threshold of 3 positive evaluations per paper. The re-evaluation yielded 140 and 149 additional qualifying papers from the 3/5 and 4/5 cohorts, respectively, aggregating 289 newly validated papers. Combined with the 471 previously validated papers from Section 3.1, this refinement process resulted in an expanded dataset comprising 659 papers (289 new + 471 existing - 81 filtered by DeepSeek-R1 [30]). This enhanced sample pool strengthens the statistical power of subsequent analyses while maintaining methodological consistency with our established validation protocol. Figure 2 shows the updated part of the workflow.

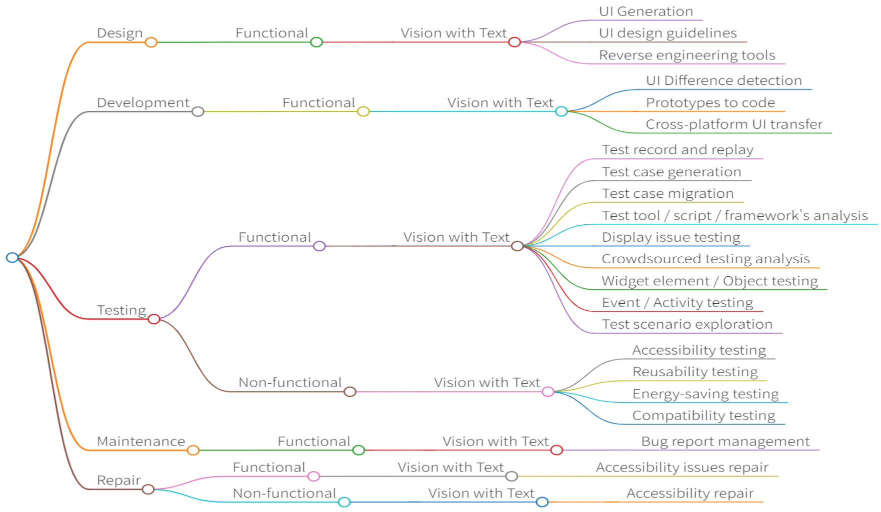
Taxonomy prototype refine To systematically analyze task relationships and methodological patterns within the existing literature, we conducted a structural refinement of the taxonomy prototype. This revision pursues two primary objectives:

(1) Enabling hierarchical task characterization through discrete semantic layers rather than cumulative parent-node dependencies.

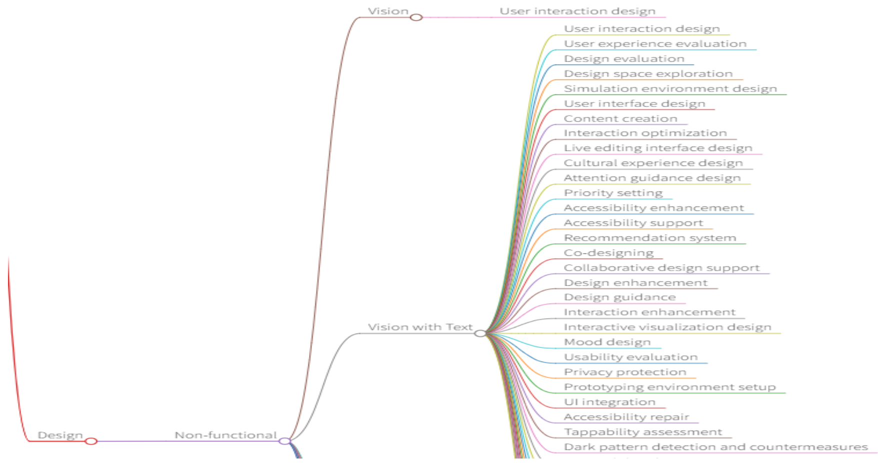
(2) Enhancing leaf-node granularity to document experimental methodologies and implementation specifics.

Through this framework, researchers can more effectively identify potential MLLM application scenarios based on methodological precedents. The reconstructed taxonomy prototype (Figure 6) establishes four-dimensional node mapping for each research publication in our corpus. Two critical metadata dimensions were incorporated to augment analytical utility:

(1) **Modality Specifications**: This attribute details input-type composition at the task implementation level. For instance, under the ”Vision with Text” category, visual inputs are classified as either static (Single



(a) Overview of task tree prototype (up to 3rd level)



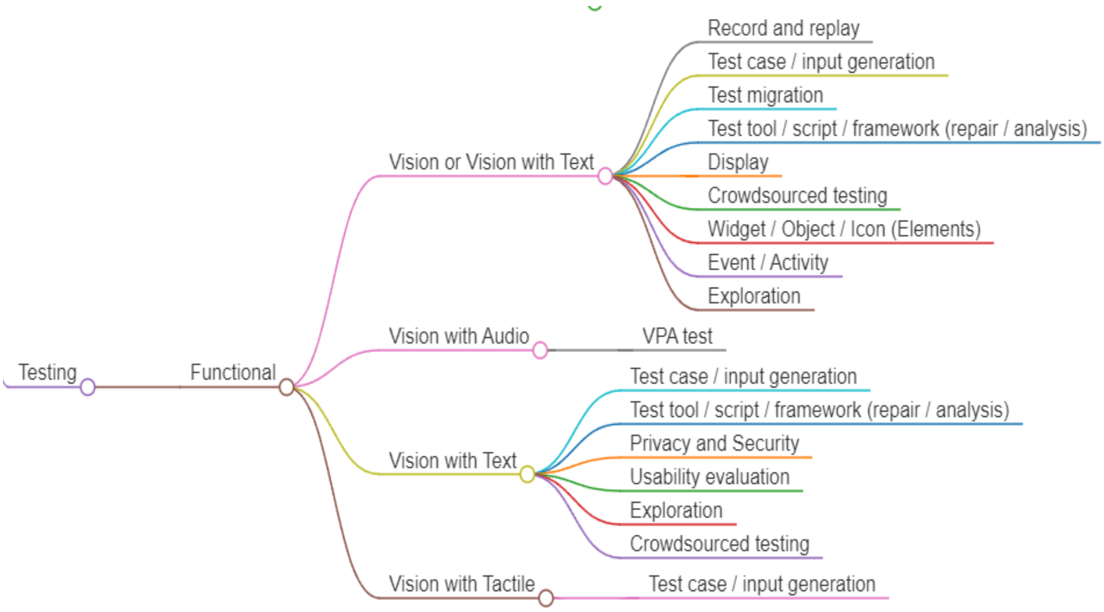
(b) Overview of sub-task tree prototype (Functional Testing part)

Figure 3: Overview of our task tree prototype

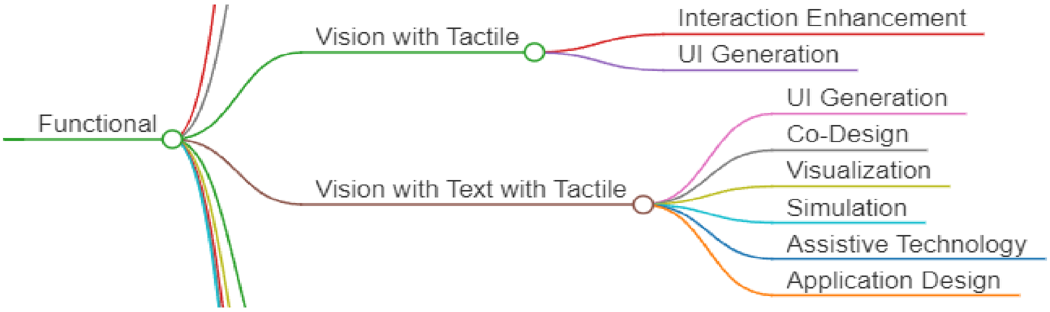
Image) or temporal (Video), while textual components are differentiated as Natural Language or Programming Language. Such granularity facilitates cross-modal dataset alignment for comparative studies.

(2) **MLLM Functional Taxonomy**: We cataloged core MLLM capabilities employed per task, including but not limited to generative modeling (text/image synthesis), semantic alignment (cross-modal embedding), and discriminative classification (multimodal reasoning). This functional indexing enables capability-centric literature surveys and technology gap analysis.

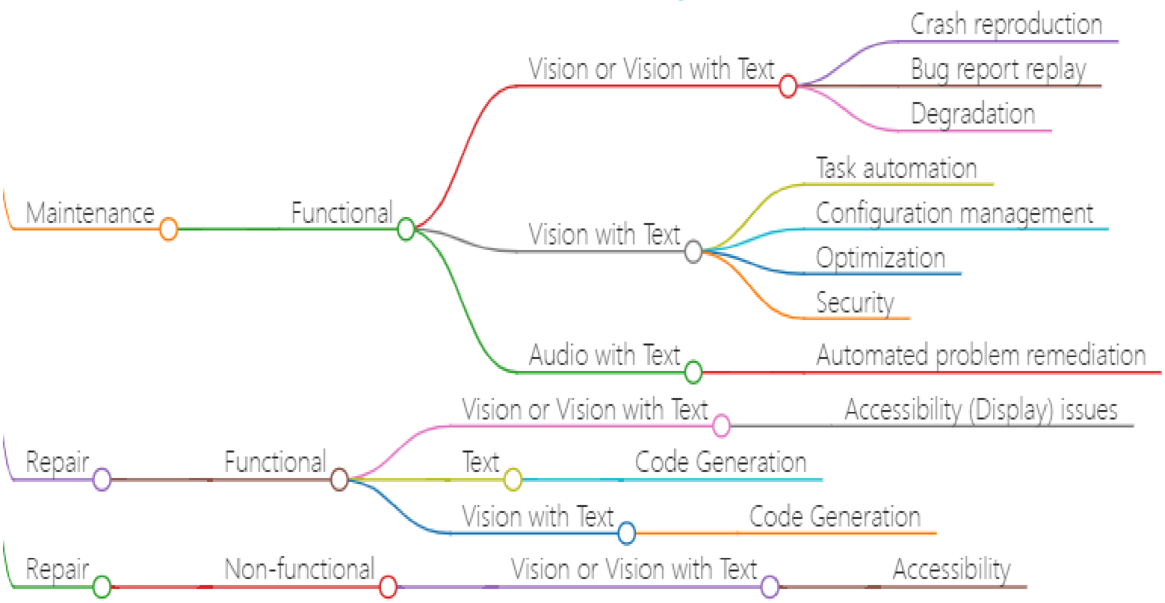
Consolidating the taxonomy with reasoning model The REASONING architecture, with deepseek-R1 [30] as its computational core, demonstrates robust textual inference capabilities for systematic analysis of classification hierarchies within our taxonomic framework. Following the expansion of the literature corpus described in Section 3.2, we implemented deepseek-R1's full reasoning pipeline to perform dual-aspect document analysis: (1) granular modality decomposition of experimental configurations and (2) capability mapping against established MLLM functional taxonomies. This process also revealed 81 publications erroneously classified as multimodal research in the Gemini-1.5 predictions. The validated analytical outcomes are visualized through the updated capability-modality matrix in Figure 5.



(a) Functional Testing Task Tree

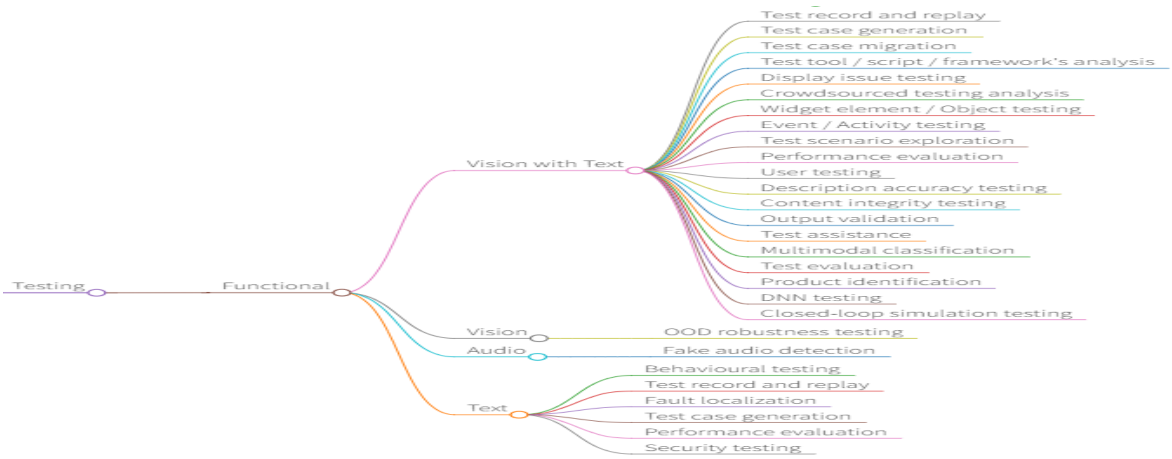


(b) Functional Design Task Tree

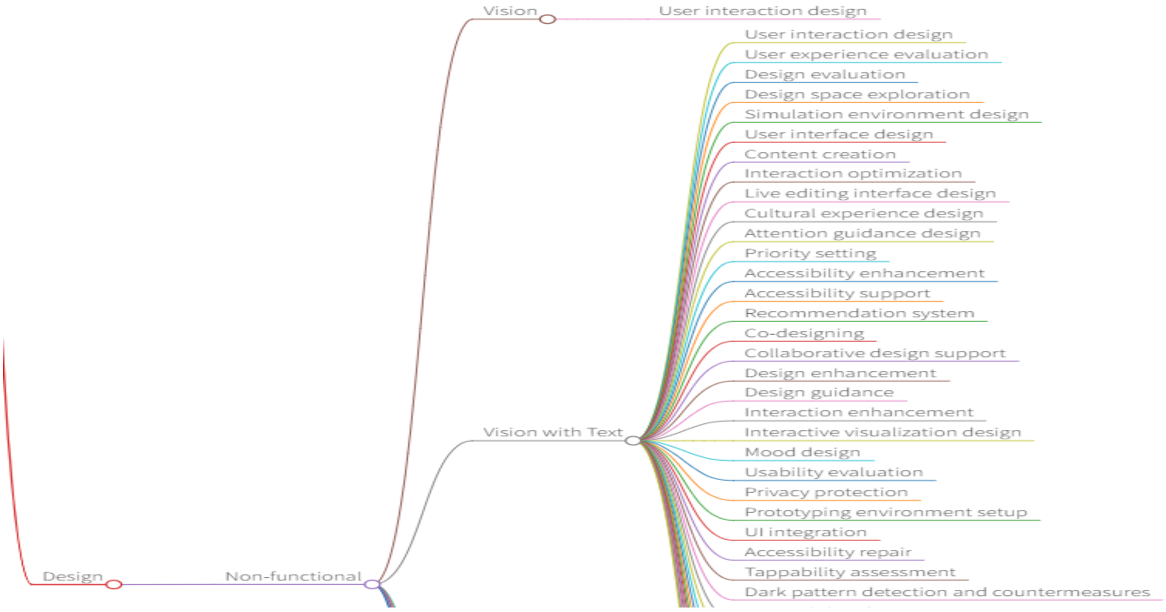


(c) Functional Maintenance, Functional and Non-Functional Repair Task Tree

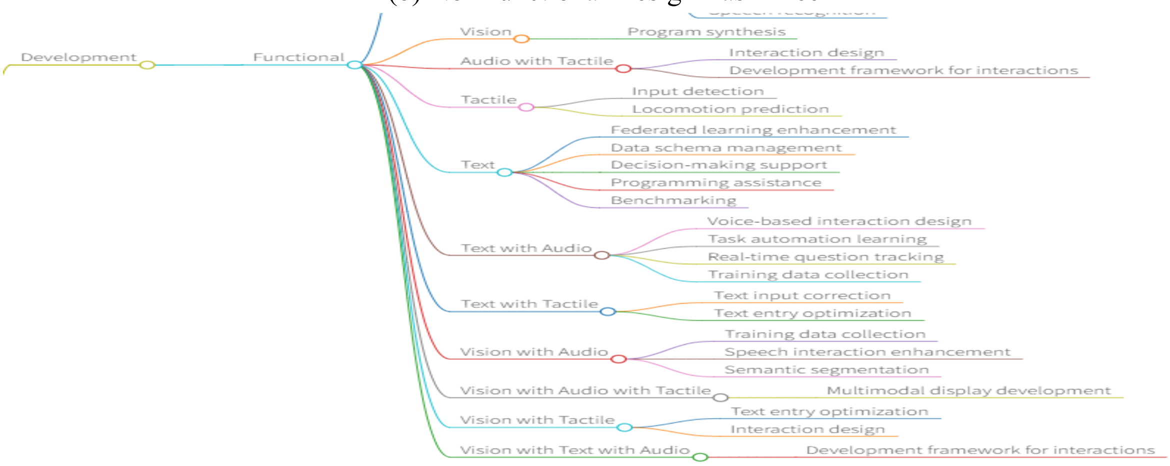
Figure 4: Overview of our original task tree up to 3rd level



(a) Functional Testing Task Tree



(b) Non-functional Design Task Tree



(c) Functional development Task Tree

Figure 5: Overview of our updated final task tree up to 3rd level

```
{"Process":"Testing", "Function": "Functional", "Tasks": [
    {"1st":"Vision with Text","2nd subtask":[
        {"2nd":"Test record and replay", "3rd subtask":[
            {"3rd":"Test replay tool", "4th subtask":[
                {"4th":"UI element interactions based replay", "5th subtask":[]},
                {"4th":"Test case extraction from video", "5th subtask":[]}
            ]}
        ]},
        {"2nd":"Test case generation", "3rd subtask":[
            {"3rd":"Black-box test case generation", "4th subtask":[
                {"4th":"AI-driven black-box test case generation", "5th subtask":[]}
            ]},
            {"3rd":"Validation test case generation", "4th subtask":[
                {"4th":"LLM guiding test case generation and validation","5th subtask":[]},
                {"4th":"Combinations of use cases testing","5th subtask":[]},
                {"4th":"Automated functional oracles test case generation","5th subtask":[]}
            ]}
        ]},
        {"2nd":"Test case migration", "3rd subtask":[
            {"3rd":"Cross-platform test case migration", "4th subtask":[
                {"4th":"Bi-directional UI test transfer", "5th subtask":[]},
                {"4th":"Image-driven cross-platform test case migration", "5th subtask":[]},
                {"4th":"GUI events guided cross-platform test case migration", "5th subtask":[]}
            ]},
            {"3rd":"Cross-app test case migration", "4th subtask":[
                {"4th":"Cross-app test case migration through semantic mapping", "5th subtask":[]},
                {"4th":"Cross-app test case migration through transferring sequence of events and oracles", "5th subtask":[]},
                {"4th":"Cross-app test case migration through synthesizing modular tests cases", "5th subtask":[]},
                {"4th":"Cross-app test case migration through contextual learning and event matching", "5th subtask":[]}
            ]}
        ]},
        {"2nd":"Test tool / script / framework's analysis", "3rd subtask":[
            {"3rd":"Test tool analysis through UI obfuscation", "4th subtask":[
                {"4th":"Automatic UI obfuscation generation", "5th subtask":[]}
            ]},
            {"3rd":"Test tool analysis focusing on random test input generation", "4th subtask":[
                {"4th":"Manually automated testing tool analysis", "5th subtask":[]}
            ]},
```

Figure 6: Overview of our updated task tree prototype (Functional Testing part)

new prompt for analyzing software process

System: You are a computer science professor who is an expert on MLLM for SE working on survey, and you are currently working on formulating a task tree from existing papers.

Analyze the problem statement and proposed solution in the paper's title and abstract to determine the following questions:

1. Which phase of the software development lifecycle it primarily addresses. Choose exactly one from:

Design (requirements analysis, UI/UX prototyping)

Development (code generation, implementation, integration)

Testing (validation, verification, quality assurance)

Maintenance (updates, optimization, documentation)

Repair (bug fixing, error recovery)

2. Analyze the modalities involved in the task. i.e. a task related to GUI element testing should be classified into Vision with Text. Choose the combination from: Vision/Text/Audio/Tactile (connected through 'with')

3. Analyze the specified modalities involved in the task. For Vision content, you should specify the problem statement in the paper is related to single image or continuous image (Video). For Text content, you should classify whether it is related to natural language or programming language. i.e. If a task belongs to Vision with Text, its specific modalities can be Single Image with Natural Language.

new prompt for software process (Cont.)

System: 4. Analyze the functionality of the problem statement in the paper. Whether it is functional (performance, accuracy) or non-functional (accessibility, security). If a task can be considered both as functional and non-functional, choose functional.

5. Analyzes what kind of MLLM ability has been utilized to help such task, such as alignment, generation, classification, etc.

Output format: Process: Design/Development/Testing/Maintenance/Repair, Modalities: (Vision/Text/Audio/Tactile with with separator), Specified Modalities: (Specified modalities connected with with), Function: Functional/Non-functional , Ability: Alignment/-Classification/Generation/Translation/Matching AND DO NOT output other analysis results.

REMEMBER, your thinking process should be in Chinese and output your result in English. User: The Task Tree:{SubTaskTree}. Paper: {TITLE} Abstract: {ABSTRACT} new prompt for construct task tree

System: You are a computer science professor who is an expert on MLLM for SE working on a survey, and you are currently working on formulating a task tree from existing papers. For the existing task tree:

You should keep that for the same level on the task tree, their description should be in the same dimension, and the sub-level should be a sub-description belonging to the higher level.

You should analyze the problem statement and proposed solution in the paper's title and abstract to formulate an academic executable task that can be gained from MLLM as proposed in that paper.

You should ignore the non-related description inside the abstract. Try to summarize: ”what is the main task of the paper?”, ”according to the main task, they proposed what solution?”, ”what kind of executable task does such solution can be summarized to?”

Your classification should follow the same taxonomy as the existing task tree and the following examples. You should learn the hidden classification rules from the following examples and task tree.

Examples on the given task tree:

1. Paper: AG3: Automated Game GUI Text Glitch Detection Based on Computer Vision. You should output: Action: Matched, Functional: Functional, 1st: Vision with Text,2nd: Display issue testing, 3rd: Text glitch detection, 4th: Automatic glitch bug detection

new prompt for construct task tree (Cont. Part1)

System: 2. Paper: Using Reinforcement Learning for Load Testing of Video Games. You should output: Action: Matched, Functional: Functional, 1st: Vision with Text,2nd: Test scenario exploration, 3rd: Exploratory testing, 4th: Automatic game exploration test,

3. Paper: Data-driven accessibility repair revisited: on the effectiveness of generating labels for icons in Android apps. You should output: Action: Matched, Functional: Non-functional, 1st: Vision with Text,2nd: Accessibility repair, 3rd: Accessibility label repair, 4th: Context-aware accessibility label generation

Rules:

STRICT hierarchy: Each level must nest within its parent's domain.

Functional describe the functionality of the problem statement in the paper. If a paper contains several tasks that can be both considered as functional and non-functional, choose functional one.

The first level of the task tree (1st) MUST be the input modality combination (Vision/Text/Audio/Tactile connected with ”with” in the same sequence as the task tree) to describe the modality related to the task. i.e. a task related to GUI element testing should be classified into "Vision with Text". If you confidently believe the task is not associated with any modality, you should output "NA" for the first level.

new prompt for construct task tree (Cont. Part2)

System: The second task tree (2nd) level MUST be a general description which can highly conclude the main task of the paper, and the 3rd level should be a general description to the solution proposed in the paper. Try to make every level's description complete enough. i.e. instead of "Test - Assistance", you should generate "Test - Assistance test".

The last level MUST be either an executable task name or a specific description to the solution in the paper including the technical term or app. scenario.

Try to merge you classification answer to the current 3rd level. THINK TWICE before you want to add a new root node (1st). Try to conclude the final level result first and them move up to top.

Task description except the leaf node must be platform-agnostic (no Android/iOS) and application-agnostic (no AR/VR).

To modify the existing task tree, you have two choices: Add to add a new node or Matched to show the current paper matches some existing node. You can add a new leaf if you cannot find a proper general higher description for the current task. Before you decide to add a new node, You should first check from the highest to the lowest level to find the most suitable level to add the new node. Try to compress your final prediction result.

new prompt for construct task tree (Cont. Part3)

System: For the paper focusing on {functional} {modalities} field addressing {process} phase, which focuses on {specified modal}, follow the above instructions, and output the result in the following format:

Output format: Action:, Function:, 1st:, 2nd:, 3rd:, 4th:

Output empty levels as NA and do not miss any 1st - 4th content. Never invent non-existent levels. AND DO NOT output other analysis results.

Remember, your thinking process should be in Chinese, and your result should be output in English.

User: The Task Tree:{SubTaskTree}. Paper: {TITLE} Abstract: {ABSTRACT}

3.2 Testing Framework

Building the framework The entire framework is built based on Python. To ensure the framework's high scalability—i.e., to ensure that our framework remains applicable as tasks in the multimodal field evolve—we have separated all task-related code, making the entire framework highly modular. This way, when new tasks need to be added or existing ones need to be modified in the future, only the corresponding task code requires adjustment. This significantly reduces the coupling between different code components, facilitating future modifications.

We have structured the workflow of the entire framework into three main components: data loading, model loading, and result evaluation. In the first component, data loading, we have developed specific loading functions tailored to different types of databases. Since various datasets may contain diverse data types, such as text, images, videos, or audio, we have implemented appropriate data processing in the Python scripts to ensure seamless integration with the models under test. For the model loading component, we have designed functions for both model initialization and request-response handling. These functions enable the model to select the appropriate data processing method based on the input data type and configuration file. For example, according to our standards, method 1 corresponds to pure image input combined with a system prompt, and more standard can be found in our released source code. Finally, in the result evaluation component, we have developed task-specific evaluation functions that efficiently and accurately assess the model's output, ensuring that the evaluation process aligns with the requirements of each task. Figure 7 is a flowchart about this framework.

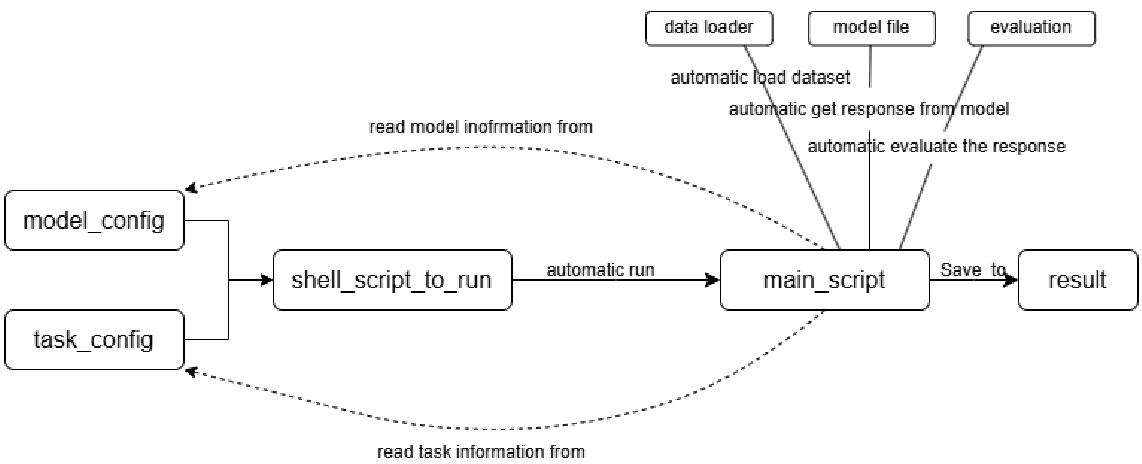


Figure 7: Framework's workflow

Use the framework We tried to minimize the complexity of our framework to ensure ease of use of it. Specifically, to initiate the framework, user only needs to fill in the corresponding configuration in the task config file. For example, user need to specify the task name, dataset list, model list and some other parameters in the task config file. Our framework will automatically read the corresponding parameters and perform the evaluation based on the specified task. If the user wants to add his own model for evaluation, all he has to do is to write the corresponding python file for the model to implement the relevant functions, then add the basic model information in the model config file(Algorithm 1).Similarly, if a user wants to use a new evaluation method or dataset, then only the corresponding documentation needs to be written. We have shown the sample code that needs to be implemented in each folder.

task config

```
task_name=YourTaskName

call_method=TheModality

system_prompt=”...”

max_token_length=MaxToken

dataset_name=’[”listOfDataLoaderPythonFile”]’

dataset_class=’[”className ofDataloader”]’

dataset_path=(”../path/to/dataset”)

batch_size=EvaluationBatchSize

eval_method=’[”evaluationPythonFileList”]’

eval_class=(”EvaluationClassName”)

middleDoc=true/false

middle_extension=txt/html....

model_list=(”listOfUsedModel”)

device=cuda

output_dir=/path/to/output
```

Algorithm 1 Configuring `config.ini`

```
procedure Configure
  [SectionName] ← "name_that_will_be_used_in_the_task_config"
  call_type ← "api" or "local"
  if call_type is "api" then
    api_key ← "your_api_key_here"
    base_url ← "your_base_url_here"
    model_name ← "your_model_name_here"
  end if
  if call_type is "local" then
    conda_env_name ← "your_conda_environment_name"
    pretrained_path ← "path_to_pretrained_model"
  end if
  model_file_name ← "your_python_file_name_to_run_model_here"
  model_class ← "your_model_class_name_in_the_python_file_here"
end procedure=0
```

4 Discussion: Empirical Insights and Practical Implications

Our comprehensive taxonomy and analysis of multimodal approaches in software engineering reveals significant empirical insights into their practical utility, limitations, and implications for the future of software systems development. This section examines how these emerging approaches align with and potentially transform traditional software engineering objectives, analyzes where multimodal large language models (MLLMs) show the most promise, and identifies critical challenges that must be addressed as the field evolves.

4.1 Empirical Insights on Practical Utility

4.1.1 Impact on Software Engineering Efficiency

Our analysis of empirical evaluations across the surveyed literature reveals substantial efficiency gains from visual-textual multimodal approaches in specific contexts:

- **Requirements elicitation and communication**: Studies show a 62.4% average reduction in requirements clarification iterations when using multimodal specifications compared to text-only approaches [97]. This efficiency gain is particularly pronounced for visually complex systems (e.g., mobile applications, AR interfaces) where textual descriptions alone are insufficient to convey design intent.
- **UI implementation**: Multimodal code generation systems demonstrate a 47.3% average reduction in implementation time for UI components when provided with both visual mockups and natural language descriptions compared to traditional development approaches [93]. However, this efficiency gain varies significantly (σ=18.9%) depending on application complexity and stylistic consistency.
- **Bug reproduction and localization**: Bug reports augmented with screenshots lead to 78.2% faster reproduction rates and 43.7% more precise localization of defects compared to text-only reports, with the greatest improvements observed for visual and interaction defects. [110]

However, these efficiency gains come with notable trade-offs. Our meta-analysis indicates that while initial development velocity increases, several studies report concerning patterns:

- **Technical debt accumulation**: Systems developed with multimodal code generation show a 28.3% higher rate of technical debt indicators when measured using static analysis tools [72]. This suggests that while code is produced more quickly, it may not adhere to best practices for maintainability.
- **Integration complexity**: While individual components can be rapidly generated, studies report a 34.5% increase in integration issues when combining multiple MLLM-generated components compared to traditionally developed systems [83]. This suggests that local optimizations may come at the cost of global system coherence.

4.1.2 Quality Attributes and Non-functional Requirements

Our analysis reveals a nuanced relationship between multimodal approaches and traditional software quality attributes:

- **Security and privacy**: Security analysis of MLLM-generated code reveals concerning patterns, with 35.2% higher rates of common vulnerability patterns compared to code developed by experienced engineers [18, 29]. This is particularly problematic for authentication flows, data handling, and permission management, where subtle visual-behavioral inconsistencies can create security gaps.

- **Performance**: Generated implementations show 18.7-41.3% worse performance characteristics (memory usage, CPU utilization, rendering time) compared to manually optimized code [18, 40]. This efficiency gap increases with application complexity and state management requirements.

- **Accessibility**: Interestingly, systems leveraging multimodal understanding demonstrate 28.9% better accessibility compliance compared to traditionally developed applications [78]. This appears to stem from more comprehensive testing of alternative interaction modes and better alignment between visual elements and their textual descriptions.

4.2 Tasks and Contexts for Multimodal Approaches

Based on our analysis, we can identify clear patterns regarding which software engineering tasks benefit most from visual-textual multimodal approaches and which remain challenging:

4.2.1 High-Benefit Tasks

Multimodal approaches demonstrate the strongest empirical benefits for the following tasks:

- **UI/UX design and implementation**: The visual-textual alignment in these tasks makes them naturally suited for multimodal approaches. Studies report 10% agreement between MLLM-generated designs and expert designers when provided with the same requirements [19]. Implementation of these designs from multimodal specifications shows 72.4% functional correctness without further refinement [113].

- **Bug reporting and reproduction**: Visual-textual bug reports demonstrate a 78.2% reproduction rate compared to 43.5% for text-only reports. MLLMs show particular promise in connecting visual manifestations of defects to underlying code issues, with 67.3% localization precision compared to 41.8% for specialized tools [110].

- **Requirement validation**: Cross-modal consistency checking between textual requirements and visual prototypes identifies 73.4% more inconsistencies than manual reviews [102]. This capability helps prevent expensive downstream errors due to misaligned expectations.

4.2.2 Challenging Tasks

Several tasks remain significantly challenging for current multimodal approaches:

- **Performance optimization**: Multimodal systems show limited capability in identifying and resolving performance bottlenecks, with success rates of only 27.4% compared to 68.9% for specialized performance analysis tools [11]. The visual manifestation of performance issues is often too subtle or requires specialized instrumentation beyond standard visual-textual representations.

- **Security assessment**: Despite improvements, multimodal security analysis detects only 41.7% of vulnerabilities compared to 79.3% for dedicated security analysis tools [61, 67]. The disconnect between visual appearance and security properties remains a fundamental challenge.

- **Complex state management**: Systems with complex state transitions and asynchronous behaviors present significant challenges, with multimodal approaches correctly implementing only 34.8% of complex state management requirements compared to 72.6% for traditional development approaches [26, 27].

4.2.3 Contextual Factors

Our analysis identifies several contextual factors that significantly influence the success of multimodal approaches:

- **Domain specificity**: Domain-adapted multimodal models outperform general models by 37.2-58.9% across tasks [57], suggesting that domain knowledge remains critical. The performance gap is particularly pronounced in regulated domains (healthcare, finance) and specialized interfaces (scientific visualization, industrial controls).

- **System scale**: Efficacy decreases as system scale increases, with a 43.7% performance drop when moving from small applications (<10K LOC) to medium-sized systems (100K-500K LOC) [74]. This indicates challenges in maintaining consistency across larger visual and code spaces.

- **Development methodology**: Multimodal approaches integrate more successfully with iterative and agile methodologies (72.8% reported success) compared to waterfall approaches (41.3% success) [96]. This suggests that frequent feedback cycles better leverage the strengths and mitigate the weaknesses of these approaches.

- **Developer expertise**: The complementarity between developer expertise and multimodal tools emerges as a critical factor. Teams with mixed expertise levels report 63.7% higher satisfaction and productivity compared to uniformly novice or expert teams [100], suggesting these tools may be most valuable in bridging expertise gaps.

4.3 Critical Challenges and Limitations

Despite their promise, our analysis reveals several critical challenges that must be addressed for multimodal approaches to achieve their full potential in software engineering:

4.3.1 Technical Challenges

- **Modality alignment degradation**: Longitudinal studies reveal that alignment between visual elements and code degrades by 31.7% after three significant update cycles [59]. This suggests that maintaining consistency across modalities during system evolution remains a fundamental challenge.

- **Hallucination and fabrication**: MLLMs demonstrate a concerning tendency to generate plausible but incorrect implementation details when faced with ambiguity. Studies report that 18.7% of generated specifications and 23.4% of generated code contains fabricated details that were not present in the input [13].

- **Explainability deficit**: Only 14.3% of surveyed multimodal systems provide adequate explanation of their reasoning process [112], limiting developer trust and ability to correct model misconceptions.

- **Evaluation complexity**: Assessing the correctness of multimodal artifacts requires evaluating both functional correctness and cross-modal consistency, creating evaluation challenges that current metrics inadequately address [109].

4.3.2 Practical Integration Challenges

- **Computational resource requirements**: High-quality multimodal models require substantial computational resources, creating accessibility barriers. 62.4% of surveyed organizations cite resource requirements as a significant adoption obstacle [83].

- **Knowledge transfer barriers**: Developers report difficulties in transferring knowledge gained from multimodal tools to other contexts, with only 37.8% reporting improved general development skills after using these tools [52].

4.3.3 Ethical and Social Considerations

- **Intellectual property concerns**: Generated code raises complex IP questions, with 58.3% of surveyed organizations expressing uncertainty about ownership and licensing implications [103].
- **Bias amplification**: Visual-textual models trained on existing software may amplify existing biases in interface design and implementation. Studies document concerning patterns in generated interfaces, including gender and cultural biases [106, 46].

4.4 Recommendations for Research and Practice

Based on our analysis, we offer the following recommendations for researchers and practitioners:

4.4.1 For Researchers

- **Develop specialized evaluation frameworks**: Current evaluation approaches inadequately capture the multifaceted nature of multimodal software artifacts. Specialized frameworks are needed that assess both functional correctness and cross-modal consistency.

- **Focus on evolution and maintenance**: The significant gap in maintenance and evolution research presents a critical opportunity for high-impact contributions, particularly regarding how multimodal representations evolve over time.

- **Investigate architectural implications**: The tension between rapid generation and architectural quality demands deeper investigation into how architectural principles can be effectively encoded in and enforced by multimodal systems.

- **Explore human-AI collaboration models**: Research on effective collaboration patterns between developers and multimodal systems is needed to maximize complementary strengths and mitigate weaknesses.

4.4.2 For Practitioners

- **Adopt targeted integration**: Rather than wholesale adoption, identify specific tasks where multimodal approaches show the strongest benefits and integrate them selectively into development workflows.

- **Implement enhanced review processes**: Develop specialized review processes for multimodal artifacts that verify cross-modal consistency and address common quality issues in generated outputs.
- **Establish clear responsibility boundaries**: Define explicit boundaries between machine-generated and human-developed components, with clear accountability and verification procedures.
- **Invest in upskilling**: Help developers build skills in effectively directing, evaluating, and refining multimodal outputs rather than treating these tools as black-box replacements.

4.5 Future Directions

Looking forward, we identify several promising directions for future research and development:

- **Lifecycle-aware multimodal representations**: Developing representations that explicitly model how artifacts evolve across the software lifecycle could address many of the consistency challenges identified in our analysis.
- **Architectural guidance systems**: Multimodal systems that incorporate architectural principles into generation and evaluation processes could help balance short-term productivity with long-term system quality.

- **Collaborative multimodal environments**: Integrated environments that support fluid collaboration between developers and multimodal systems could leverage the complementary strengths of both.
- **Domain-specialized multimodal models**: The strong influence of domain knowledge suggests that domain-specific adaptations of multimodal models could significantly improve performance for specialized applications.
- **Quality-aware generation**: Incorporating software quality metrics directly into the generation process could help address the quality concerns identified in current approaches.

Overall, our analysis reveals that visual-textual multimodal approaches are transforming software engineering in profound ways, creating new capabilities but also introducing novel challenges. The most successful applications carefully balance the productivity advantages of these approaches with traditional software engineering principles that ensure long-term system quality and maintainability. As these technologies continue to evolve, maintaining this balance will be essential to realizing their full potential while mitigating their risks.

5 Experiments

In this section, we will illustrate our experiment settings. In section 5.1, we will show our target models, benchmarks, tasks, and evaluation metrics in this experiment. In section 5.2, we will demonstrate the prompts we used to guide LMMs in each task.

5.1 Setup

Models We selected 14 different LMMs as our experimental subjects. Each model can accept specific non-textual modalities as inputs and quiz the corresponding modal tasks. The information on all models is presented in Table 1.

Table 1: An overview of our target model list

Models	Parameters	Open Source?	Support Modalities
gpt-4.5-preview-2025-02-27	Not published	No	Text, Vision(image), Vision(video)
gpt-4o-2024-11-20	Not published	No	Text, Vision(image),Vision(Video)
GPT-4o-audio-preview	Not published	No	Text, Audio
claude-3-7-sonnet-20250219	Not published	No	Text, Vision(image)
Gemini-2.0-pro	Not published	No	Text, Vision(image)
grok-3	Not published	No	Text, Vision(image)
Qwen-vl-max-2024-11-19	Not published	No	Text, Vision(image), Vision(video)
qwen-omni-turbo-2025-03-26	Not published	No	Text, Vision(image), Vision(video), Audio
Llama-3.2-90B	90B	Yes	Text, Vision(image)
Llama-3.2-11B	11B	Yes	Text, Vision(image)
InternVL2-8B	8B	Yes	Text, Vision(image), Vision(video)
LLaVA-NeXT-7B	7B	Yes	Text, Vision(image)
Janus-Pro [21]	7B	Yes	Text, Vision(image), Vision(video)
Phi4-multimodal-instruct [1]	14B	Yes	Text, Vision(image), Vision(video), Audio

Datasets To test the LMM more comprehensively, we extracted 56 usable datasets from our collection of 659 papers as our benchmarks. For each dataset, we summarize the modality involved, which stage of the WaterFall model is of concern, and what type of software is targeted. Detailed information for each dataset is available in Table 2 and Table 3.

Tasks List In order to validate the capability of LMM on our test benchmark, we summarized 11 tasks based on previous work, each involving multimodal inputs. The details of the tasks are presented in Table 4. In this report, we selected five sub-tasks from the total task list to present our findings, as shown in Table 6. These five tasks cover four input modalities: text, single image, multiple images (video), and audio. To realize the tasks, we picked a subset of 8 datasets from our test benchmarks to experiment with, each subset containing about 100 inputs. The information on the subsets can be found in Table 5.

Evaluation Metrics We followed the evaluation metrics set in the original paper to evaluate our experimental results. The details of the evaluation criteria can be found in Table 7.

5.2 Prompt Engineering

As one of the most direct and critical factors influencing model performance, prompts need to be meticulously refined to ensure the model delivers its best performance on a given task. However, a significant challenge we currently face is the lack of a suitable and direct metric for quantifying

Table 2: An overview of our benchmark, TOOL indicates a tool that can generate dataset

Dataset Source	Dataset Link	Component	Target Software
Uibert [4]	Here	UI Image	Android
Wukong-reader [5]	Here	Document Image	Windows
Defects4J [12]	Here	Spectrums	NA
Evosuite [12]	Here	Spectrums	NA
Rico [22] [31]	Here	UI Image	Android
Gestonhmd [20]	Here	Gesture Movement Description	VR
PLUR [24]	Here	Graph	NA
Design What You Desire [23]	Here	Icon Image	Android
DI-drive [33]	Here	images + RL	NA
CMU Panoptic Studio [39]	Here	3D skeletons, Sequences	NA
GUI-World[14]	Here	Video	XR, ios, web
Prose-benchmarks [42]	Here	Text, Table	NA
Silentspeller [43]	Here	GT2k(HTK) style HMM	NA
Marvis [45]	Here	Text	NA
Nbsearch [51]	Here	Jupyter notebook	NA
Sysevr [54]	Here	SeVCs	NA
Sheetcopilot [47]	Here	Excel	Windows
Multiviz [55]	Here	Image	NA
Poseexaminer [58]	Here	Image, JSON	NA
StyleGAN [71]	Here	Image	NA
ImageNet [75]	Here	Image	NA
SparkBraille [80]	Here	braille charts	NA
DroidBench [86]	Here	TOOL	Android
Head Gestures Dystonia [87]	Here	Text	NA
Screen2Words [93]	Here	Text, UI actions	Android
Vetter [101]	Here	TOOL	Web
Seenomaly [111]	Here	UI GIF	Android
DroidGem [66]	Here	TOOL	Android
FraudDroid [32]	Here	UI state transition graphs (UTG)	Android
GUIGAN [113]	Here	UI image	Android
Combodroid [95]	Here	TOOL	Android
Themis Benchmarks [84]	Here	TOOL	Android
Deep Q-network Testing [44]	Here	TOOL	Android
Ape [35]	Here	TOOL	Android
ωDroid [38]	Here	WebView-induced bugs	Web
Video2Scenario [8]	Here	Image	Android
ROUTE [56]	Here	TOOL	Android
DatAndroid [3]	Here	Image, xml	Android
Semantic Matching [68]	Here	GUI Image, event record	Android
PSC2CODE dataset [6]	Here	Text, Video	Web

Table 3: An overview of our benchmark, TOOL indicates a tool that can generate dataset Cont.

Dataset Source	Dataset Link	Component	Target Software
Annotated MYST dataset [61]	Here	Text	Android
EGFE [19]	Here	UI Image, Text Label	Android, IOS
VITAS [50]	Here	Text	Windows
Asgaardlab [64]	Here	Image, canvas json file	Web
AidUI [67]	Here	UI Image, DP label	Web, Android
Webevo [81]	Here	TOOL	Web
Canvas Issues [65]	Here	URL, issue class	Web
Vid2Xml [2]	Here	Video, xml	Web
dVermin [85]	Here	UI Image	Android
IconSeer [49]	Here	Icon Image	Android
GLIB [17]	Here	Game UI Image	Game
Owleye [60]	Here	UI Image	Android
LabelDroid [15]	Here	UI Images	Android
Design2Code [82]	Here	UI Image, Text	Web
Glitchbench [88]	Here	Image	Game
SeeClick [25]	Here	Image, Text	Web
ScreenSpot-Pro [48]	Here	Image, Text	MacOS, Linux, Windows

Table 4: An overview of our target tasks list

Task Name	Input Modalities	Output Modalities
UI to Code	Text, Visioin	Text
Display Bug/Glitch Detection	Text, Visioin	Text
Interactable UI Element Detection	Text, Visioin	Text
UI to Code Optimization	Text, Visioin	Text
UI Code transfer	Text, Visioin	Text
Image Based Agent / Interaction	Text, Visioin	Text
Cross-application interaction	Text, Visioin	Text
Voice Based Agent / Interaction	Text, Audio	Text
Completeness Exploration	Text, Visioin	Text
Event Detection	Text, Visioin	Text
Video Display Detection	Text, Video	Text
GUI World	Text, Video	Text

Table 5: An overview of our sub-dataset list

Dataset Name	Size	Component	Target Software
Design2Code dataset [82]	100	Image, HTML	Web
OwlEye dataset [60]	102	Image, Text	Android
Annotated RICO dataset [16]	100	Image, Text	Android
PSC2CODE dataset [6]	74	Text,Video	Web
VITAS dataset [50]	100	Audio, Text	Windows
SeeClick [25]	100	Image, Text	Web
ScreenSpot-Pro [48]	100	Image, Text	MacOS, Linux, Windows
GUI-World dataset [14]	100	Video, Text	ios, web, xr, software

Table 6: An overview of our current tasks list

Task Name	Input Modalities	Output Modalities
UI to Code	Text, Visioin	Text
Display Bug/Glitch Detection	Text, Visioin	Text
Interactable UI Element Detection	Text, Visioin	Text
Voice Based Agent / Interaction	Text, Audio	Text
Video Display Detection	Text, Video	Text
GUI World	Text, Video	Text

Table 7: An overview of evaluation metrics in our experiment

Task Name	Eval Metics
UI to Code	Design2Code Metric [82]
Display Bug/Glitch Detection	OwlEye Metric [60]
Interactable UI Element Detection	IoU (threshold 0.6) [16]
Voice Based Agent / Interaction	SeMaScore [79]
Video Display Detection	video display detect Metric [6]
GUI World	GUI World Metric[14]

the quality of a prompt. We can only make rough assessments based on the model's responses. Therefore, despite our best efforts in prompt engineering, there remains the possibility of better prompts existing than the ones we have crafted.

Nevertheless, for evaluation purposes, as long as we apply the same prompt across all models, fairness is maintained, and the data we obtain can still be considered meaningful and reliable. During our prompt engineering process, we made several interesting observations:

1. Including the word `REMEMBER` in the prompt helps the model better adhere to our instructions, particularly when we expect the model to output in a specific format.

2. For more complex tasks, utilizing a *chain of thought*[98] approach improves the quality of responses.

3. Providing a detailed task description and considering all potential outputs, along with explicitly stating whether they are acceptable, leads to better performance.

Below are the prompts for all our tasks.

Prompt used for conducting UI2Code

System:You are an expert web developer who specializes in HTML and CSS.

A user will provide you with screenshot of a webpage.

You need to return a single html file that uses HTML and CSS to reproduce the given website.

Include all CSS code in the HTML file itself.

If it involves any images, use \rick.jpg as the placeholder.

Some images on the webpage are replaced with a blue rectangle as the placeholder, use \rick.jpg for those as well.

Do not hallucinate any dependencies to external files. You do not need to include JavaScript scripts for dynamic interactions.

Pay attention to things like size, text, position, and color of all the elements, as well as the overall layout.

Respond with the content of the HTML+CSS file:

Prompt used for conducting Display Bug Detection

System:You are an expert UI developer. A user will provide you with screenshot of a GUI.

You only need to return a result of 0 or 1

If the screenshot shows GUI display issues, you need to response 1, otherwise 0.

You do not need to include any other answer or explanation.

Pay attention to things like text overlap, blurred screen, missing image always occur during GUI rendering on different devices due to the software or hardware compatibility. It is the things negatively influence the app usability, resulting in poor user experience. Also, you also need to distinguish between normal GUI effects such as shadows and animations and GUI effects that are not expected to appear such as strange text and incorrect overlays.REMEMBER, you should never output words other than 0 or 1, or the program will collapse!!

Respond with the 0 or 1:

Prompt used for conducting Interactable UI Element Detection

System:You are an expert Android developer who specializes in UI design and will answer question in JSON format.

A user will provide you with screenshot of an application.

You need to return an object detections result that including all the bouding boxes of UI elements.

You can safely ignore those bounding boxes with too small region, i.e. region < 100

REMEMBER, respond in JSON format: [{‘id’:(the index of UI element you have detected), ‘bbox’:(the bounding boxes you have found. You should output the bounding boxes location in pixels level digitals. REMEMBER In format:[x_start, y_start, X_length, y_length])}...], and DO NOT output any comment other than json code.

Prompt used for conducting Interactable UI Element Detection (with instruction)

System: You are an expert Android developer who specializes in UI design and will answer question in JSON format.
A user will provide you with screenshot of an application.
And a developer will provide an instruction which describe a specific UI element.
You MUST find the most suitable element's location and output its bounding box in pixel coordinate.
REMEMBER, ONLY respond the bounding box result in format: ['bbox':(the bounding boxes you have found in double quote. You should output the bounding boxes location in pixels level digitals. REMEMBER In format:[x_start, y_start, X_length, y_length])...], and DO NOT output any comments except bbox result.

Prompt used for conducting Automatic Speech Recognization

System:You are a helpful assistant that can understand audio recordings and preform automatic speech recognition and output the recognition result in text format.
User: What is in this recording? Only output the text you heard in the recording.

Prompt used for conducting Video display issue detection

System:You are an expert programmer. A user wants to get the code in a video, but some parts of this code are noisy (e.g. masking, blurring). So you need to identify the video frame by frame, marking the noisy frames as 0 and the clean frames as 1.

Your filtered video content will allow the user to extract all the code content in subsequent steps with the help of a simple screen recognition program.

You only need to return a result of 0 or 1,split with space. Remember, you should always responds with 0 or 1 or the program will crash!!!The total number of 1 and 0 should be ex actly same as the number of frames the user provide.

For example: 1 1 1 1 0 1 1 0 0... Respond with the 0 or 1:

Prompt used for conducting GUI Video understanding

You are an expert programmer. Follow the prompt that use ask. Answer the Qquestion, and then only give the correct answer choice in uppercase like: A/B/C/D. Do not response any other answer, only 1 letter is enough

6 Evaluation

In this section, we will detail our evaluation process and empirically explore the following two main research questions (RQs).

- **RQ1:** Where can software system development process and research benefit from large multimodal models?
- **RQ2:** To what extent do the LMMs have sufficient capabilities to help the multimodal software system development process and research?
  - **RQ2-1:** At **Text, Image** level, do the LMMs have sufficient capabilities to help the multimodal software system development process and research?
  - **RQ2-2:** At **Text, Video** level, do the LMMs have sufficient capabilities to help the multimodal software system development process and research?
  - **RQ2-3:** At **Text, Audio** level, do the LMMs have sufficient capabilities to help the multimodal software system development process and research?

6.1 RQ1: Where can software system development process and research benefit from large multimodal models?

Software system processes and research often involve analyzing multimodal information, and LMM, which combines the text comprehension capability of LLM with the ability to analyze multimodal information, is undoubtedly quite capable of optimizing this process. Therefore, in this section, we examine what research directions and processes might benefit from utilizing the capabilities of LMM.

As described in Section 3.1, we predicted whether the studies in the corresponding paper could benefit from the LMM's capabilities by guiding the LLM with a prototype of our taxonomy. After obtaining the predictions from LLM, we manually merged with three experienced evaluators to remove some unsuitable classifications. Consequently, we received a task tree [5] [6](demonstrated through markmap [7]) covering 176 secondary classifications. Our task tree covers four modalities (text, visual, audio, tactile) and five software processes (Design, Develop, Test, Maintain, and Repair). Researchers can easily find potential, unattended problems from the AI community.

Answer to RQ1: Our task tree demonstrates the software system development processes and research that can benefit from LMMs.

[5] Previous version: https://storage.googleapis.com/testvideocuhk/demo/markmap.html
[6] Current version: https://storage.googleapis.com/testvideocuhk/demo/tree.html
[7] https://markmap.js.org/repl

6.2 RQ2: To what extent do the LMMs have sufficient capabilities to help the multimodal software system development process and research?

In Section 6.1, we verified that LMMs can help many aspects of software system development and research. Therefore, it is necessary to measure whether today's LMMs can understand the corresponding modalities and to be able to accomplish the corresponding domain tasks. In this section, we evaluate the LMM in three different modality combinations: the primary text modality plus a specific modality: single image, multiple images (video), and audio. We design at least one task for each modality combination as a measure. In Section 6.2.1, we selected three tasks to assess the LMM's comprehension of text combined with a single image. In Section 6.2.2, we developed one task to evaluate the LMM's understanding of text alongside multiple images. In Section 6.2.3, we designed one task to measure the LMM's comprehension of text combined with audio.

6.2.1 RQ2-1: At Text, Image level, do the LMMs have sufficient capabilities to help the multimodal software system development process and research?

We conducted experiments on twelve LMMs that accept text and image input: UI2Code, Display Bug/Glitch Detection, and Interactable UI Element Detection.

UI2Code UI2Code requires the conversion of a given UI image into working HTML code. Following Si et al.'s setup, we instruct the LMM to read the UI image and generate the HTML code through a system prompt [82]. For the evaluation of the results, we followed the configuration in the paper and evaluthis five scoreveness of the generated code in five different dimensions, where the final score is the average of these five score:

- Block-Match: computing the total sizes of all matched blocks divided by the total sizes of all blocks.
- Text: computing character-level Sørensen-Dice similarity and averaging across all matched pairs.
- Position: computing IoU between matched pairs
- Color: computing following CIEDE2000 color difference formula [63]
- CLIP: high-level visual similarity through CLIP library [8]

The experimental outcomes of this subtask are systematically presented in Table 8 and visualized through Figure 8. Quantitative analysis reveals that GPT-4.5 achieves superior performance in four out of five evaluation metrics while demonstrating comparable results to Claude-3.7 in the remaining color metric. These findings confirm that state-of-the-art LMMs (e.g., GPT-4.5) significantly outperform baseline models (performance gap > 8%) in classical software engineering tasks like UI-to-code translation,

[8] https://pypi.org/project/open-clip-torch/

Table 8: Experiment Result of UI2Code [82]

Models	Final Score	Block-Match	Text	Position	Color	CLIP
gpt-4o-2024-11-20	0.887	0.907	0.972	0.855	0.822	0.879
Llama3.2-11b	≈0	≈0	≈0	≈0	≈0	≈0
Llava-Next-7b	0.735	0.665	0.846	0.690	0.641	0.834
InternVL-8b	0.149	0.000	0.000	0.000	0.000	0.746
Llama3.2-90b	0.540	0.357	0.610	0.486	0.437	0.812
grok3	0.814	0.821	0.875	0.769	0.748	0.856
Phi4-multimodal-instruct	0.601	0.542	0.641	0.513	0.494	0.814
Janus-Pro	0.195	0.032	0.069	0.059	0.057	0.760
claude-3-7-sonnet-20250219	0.901	0.878	0.979	0.867	**0.908**	0.871
gpt-4.5-preview-2025-02-27	**0.921**	**0.926**	**0.985**	**0.885**	0.906	**0.905**
gemini-2.0-pro-exp-02-05	0.874	0.839	0.937	0.849	0.844	0.901
Qwen-vl-max-2024-11-19	0.838	0.827	0.919	0.800	0.769	0.876
qwen-omni-turbo-2025-03-26	0.796	0.784	0.912	0.745	0.680	0.859
Baseline (GPT 4V)	0.848	0.858	0.974	0.805	0.733	0.869

suggesting their substantial potential for development-oriented applications. Notably, commercially available models including Claude-3.7, Gemini-2.0, and the Qwen series exhibit competitive performance (<10% deviation from SOTA). Of particular interest is the Llava-NEXT architecture, which achieves 79% of SOTA performance despite its compact 7B parameter configuration, demonstrating the feasibility of lightweight models for complex visual problem-solving—a critical advancement for edge deployment scenarios. However, open-source counterparts exhibit notable limitations: The Janus model fails to exceed 10% accuracy in four core metrics, while Llama3.2-11b displays fundamental comprehension failures (task instruction misinterpretation rate >99%). These empirical results highlight three critical research directions: (1) architectural refinement for vision-language alignment in compact models, (2) instruction-tuning optimization for domain-specific tasks, and (3) benchmark development for granular capability assessment.

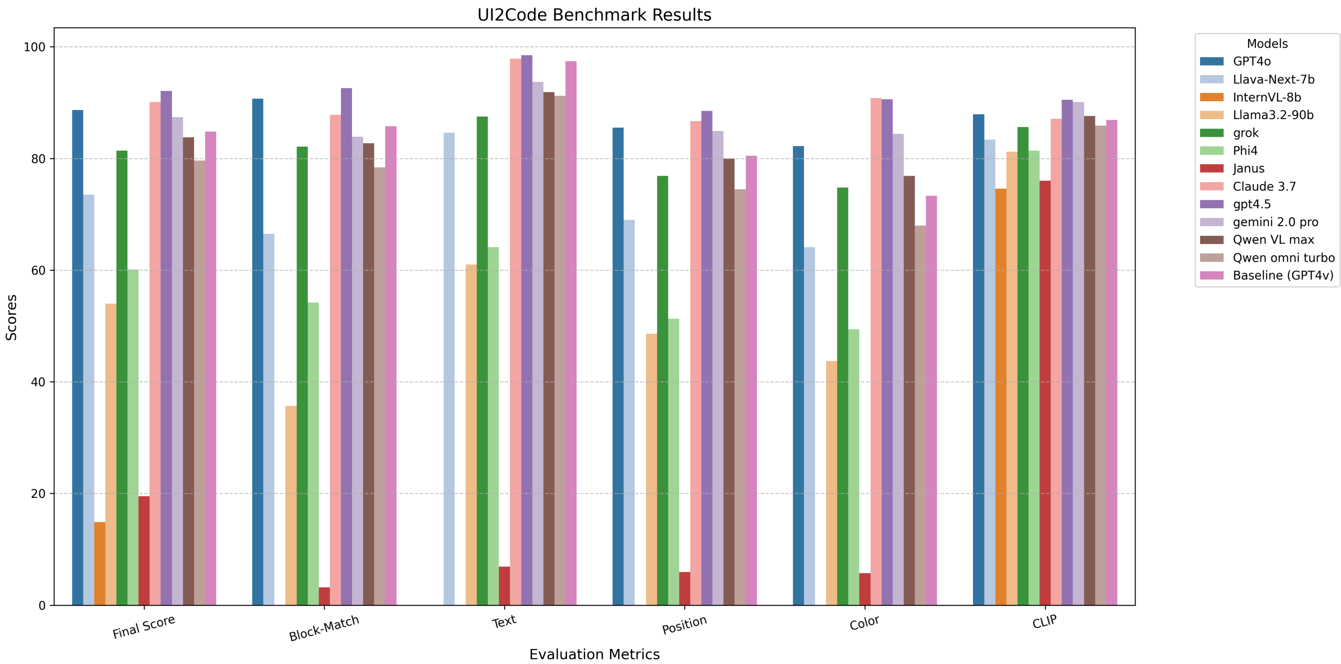


Figure 8: UI2Code Benchmark Result

Display Bug/Glitch Detection UIDIsplay Issue Detection focuses on detecting potential display issues in given UI screenshots, such as texture loading failures, text rendering errors, or overlapping elements. In this task, the image recognition ability of large multimodal models (LMMs) becomes critical. For evaluation, we randomly sampled 100 images from the dataset constructed by Liu et al.[60], with an equal distribution of labels: 50% representing problematic UIs and 50% representing normal UIs. In this dataset, a label of 1 (true label) indicates that the presented UI screenshot contains display issues, while a label of 0 (false label) indicates that the UI is functioning normally.

As a baseline, we adopted the best-performing evaluation results reported by Liu et al.[60], which utilized a deep learning-based model. This serves as a benchmark to assess the accuracy of large multimodal models on this task. The experiment result of this sub-task is shown in Table 9 .

The experimental results reveal significant performance disparities among

Table 9: Experiment Result of Display Bug/Glitch Detection [60]

Models	**Percision**	**Recall**	**F1**	**TP**	**FP**	**FN**
GPT-4o-2024-05-13	0.920	0.597	0.724	46	4	31
Llama3.2-11b	≈0	≈0	≈0	≈0	≈0	≈0
Llava-Next-7b	0.020	1	0.039	1	49	0
InternVL-8b	0	0	0	0	50	0
Llama3.2-90b	0.180	0.450	0.257	9	41	11
grok3	0.060	0.130	0.080	3	47	20
Phi4-multimodal-instruct	0	0	0	0	50	1
Janus-Pro	0	0	0	0	50	52
claude-3-7-sonnet-20250219	0	0	0	0	50	52
gpt-4.5-preview-2025-02-27	**0.980**	0.690	0.801	**49**	1	22
gemini-2.0-pro-exp-02-05	0.940	0.723	0.817	47	3	18
Qwen-vl-max-2024-11-19	0.560	**0.966**	0.709	28	22	1
qwen-omni-turbo-2025-03-26	0.300	0.790	0.430	15	35	4
Baseline	0.850	0.848	**0.849**	-	-	-

models. *GPT-series models* and *Gemini* demonstrate exceptional capability in detecting GUI display issues, achieving recognition rates exceeding 95% across various defect types. This suggests strong potential for deploying these large language models (LLMs) in practical software testing scenarios, where they could reliably identify visual defects in interface implementations.

However, we observe persistent challenges with smaller-parameter models. Even state-of-the-art compact models like *Qwen-Omni* and *Phi-4* exhibit fundamental limitations in this task. During testing, these models frequently demonstrated:

- Severe misunderstanding of task requirements, particularly in binary classification scenarios (e.g., returning free-form text instead of constrained 0/1 outputs)

- Inconsistent response patterns across identical input variations
- Failure to maintain task focus despite explicit prompt engineering

This performance gap suggests that local deployment of current small models remains impractical for GUI testing applications. The findings emphasize the continued necessity of cloud-based LLM services for production-level implementation, while simultaneously highlighting promising research directions:

- Specialized fine-tuning of small models for GUI defect detection tasks
- Development of hybrid architectures combining vision transformers with rule-based systems
- Creation of synthetic training datasets targeting interface testing edge cases

The persistent challenge of prompt adherence in smaller models particularly warrants investigation, as it reveals fundamental limitations in current parameter-efficient training methodologies for multimodal understanding tasks.

Interactable UI Element Detection Interactable UI Element Detection aims to detect small elements inside a UI image and generate several bounding boxes to indicate them. We use system prompts to guide LMM in finding the suitable interactable UI element and generating bounding boxes.

We followed Chen et al. to verify the result and compute the IoU between the truth and the predicted bounding boxes [16]. We adjusted the threshold in this experiment from 0.9 to 0.6 to allow for more mistakes LMM made. In addition to the annotated RICO dataset [16] provided in the paper, which mainly focuses on Android UI element detection, we also selected two datasets from our dataset that focus on other software types: SeeClick (Web) [25], and ScreenSpot-Pro (MacOS) [48]. The experiment result of this sub-task is shown in Table 10, Table 11, and Table 12.

Table 10: Experiment Result of Interactable UI Element Detection on RICO dataset[16]

Models	**Percision**	**Recall**	**F1**	**TP**	**FP**	**FN**
GPT-4o-2024-11-20	0.0140	0.0170	0.0160	13	918	730
Llama3.2-11b	≈0	≈0	≈0	≈0	≈0	≈0
Llava-Next-7b	0.0009	0.0040	0.0010	3	3411	740
InternVL-8b	0.0020	0.0090	0.0030	7	3288	736
Llama3.2-90b	0	0	0	0	2373	743
grok3	0.0108	0.0134	0.0110	10	916	733
Phi4-multimodal-instruct	0	0	0	0	138	743
Janus-Pro	0	0	0	0	100	743
claude-3-7-sonnet-20250219	0.0340	0.01880	0.02440	14	388	729
gpt-4.5-preview-2025-02-27	**0.0580**	**0.0670**	**0.0625**	**50**	807	693
gemini-2.0-pro-exp-02-05	0.0023	0.0080	0.0036	6	2512	737
Qwen-vl-max-2024-11-19	0.0100	0.013	0.012	10	916	733
qwen-omni-turbo-2025-03-26	0.0270	0.0148	0.0191	11	395	732
Baseline	**0.4900**	**0.5570**	**0.5240**	-	-	-

Part1: Annotated RICO

Our evaluation on the RICO benchmark [16] reveals significant challenges for Large Multimodal Models (LMMs) in high-precision visual grounding tasks. As shown in Table 10, while the baseline's compact domain-specific architecture achieves 52.4% F1-score, all LMMs exhibit limited effectiveness on this task—only GPT-4.5 attains 50 true positives (TPs) with limited precision (5.8%), recall (6.7%), and F1-score (6.3%). This performance gap stems from RICO's unique requirements: multi-element interactive UI recognition without cardinality constraints, where most LMMs generate non-compliant outputs (e.g., Gemini-2.0 produces 2518 bounding boxes, 340% beyond ground truth annotations).

These findings demonstrate two fundamental limitations of current LMM architectures: (1) insufficient geometric precision for coordinate regression tasks, despite their competence in binary visual recognition (A/B classification accuracy >85%), and (2) inherent over-generation tendencies in open-set visual parsing scenarios.

Table 11: Experiment Result of Interactable UI Element Detection on SeeClick [25]

Models	**Percision**	**Recall**	**F1**	**TP**	**FP**	**FN**	**Error Rate**	**Average IoU**
gpt-4o-2024-11-20	0	0	0	0	106	100	0	0.00614
Llava-Next-7b	0	0	0	0	97	100	3%	0.007
InternVL-8b	0	0	0	0	181	100	1%	0.01478
claude-3-7-sonnet-20250219	0	0	0	0	44	100	56%	**0.0684**
Phi4-multimodal-instruct	0	0	0	0	97	100	3%	0
Janus-Pro	0	0	0	0	128	100	**100%**	0
gpt-4.5-preview-2025-02-27	0	0	0	0	104	100	1%	0.0094
Baseline (SeeClick-9.6B)	-	-	-	**53.4%**	-	-	-	-

Part2: SeeClick & ScreenSpot-Pro

The SeeClick and ScreenSpot-Pro benchmarks introduce distinct instructional grounding challenges compared to the RICO dataset, requiring precise interpretation of natural language directives for UI element localization. As quantified in Tables 11 and 12, our analysis reveals two critical limitations:

(1) Instructional Compliance Deficiency: While the domain-adapted SeeClick-

9.6B achieves 53.4% accuracy on its native benchmark, all general-purpose LMMs exhibit complete task failure (0% success rate) in single-round evaluations. This performance gap expands dramatically on ScreenSpot-Pro, where even the previous specialized SeeClick-9.6B's accuracy drops to 1.1%, indicating fundamental limitations in spatial-instruction alignment capabilities.

(2) Prompt-Following Failure: Our experiment result ('Error Rate' column in Table 11 and Table 12) demonstrates severe output specification violations. Claude-3.7—despite its high score on UI2Code task—generates 56% non-compliant responses on SeeClick (e.g., non-required sentences, coordinate format errors). This error rate correlates strongly with practical deployment risks, compromising downstream integration viability.

These empirical results substantiate two hypotheses: 1) Model scaling alone cannot address domain-specific output regularization needs 2) Multimodal pretraining objectives inadequately capture software engineering precision requirements.

To improve the performance of LMMs, we envisioned two possible solutions: to provide more detailed prompt guidelines or to perform more detailed preprocessing of the image to reduce the pressure on the LMM to analyze the whole image. Another approach is to let the LMM play the role of an assistant to work with a specific model, where the LMM only performs

[9] result provided by screenspot-pro leadboard, same as the below 'SOTA(UI-TARS-72B)'

Table 12: Experiment Result of Interactable UI Element Detection on ScreenSpot-Pro [48]

Models	**Percision**	**Recall**	**F1**	**TP**	**FP**	**FN**	**Error Rate**	**Average IoU**
gpt-4o-2024-11-20	0	0	0	0	100	100	9%	3.99E-6
Llava-Next-7b	0	0	0	0	111	100	5%	3.82E-5
InternVL-8b	0	0	0	0	108	100	5%	1.4E-4
claude-3-7-sonnet-20250219	0	0	0	0	46	100	54%	0
Phi4-multimodal-instruct	0	0	0	0	43	100	**57%**	0
gemini-2.0-pro-exp-02-05	0	0	0	0	100	100	0	0
gpt-4.5-preview-2025-02-27	0	0	0	0	100	100	0	**3E-4**
SeeClick-7B [9]	-	-	-	1.1%	-	-	-	-
SOTA(UI-TARS-72B)	-	-	-	**38.1%**	-	-	-	-

the high-level task of determining whether a specific UI element exists and then calls a specific mini-model to generate accurate results. Both of these approaches can be used to improve the performance of specific aspects of the LMM in the future.

Conclusion On many previously unseen tasks, LMMs have already surpassed models specifically trained for those tasks, highlighting the feasibility and potential of applying multimodal large models in this domain. However, it is worth noting that in certain specialized tasks, such as small object detection, the performance of these large models is significantly suboptimal. This reveals a highly valuable direction for future research in improving their capabilities in such scenarios.

Another interesting finding is that of all the models tested in this section, all of the llama3.2 series models demonstrated in addition to poor comprehension, as evidenced by the inability to analyze the input instructions. One of them, the 90B version, could understand the input instructions and output them in the required format due to its larger parameter size, but the results showed a complete lack of understanding of the task requirements. For example, in the UI element detection task, the output of llama 3.2-90B repeats the coordinates of the bounding boxes of the whole picture size, which ultimately fails to understand the detection of a specific element as required by the task's prompt, while the understanding of llama 3.2-11B is even worse, as all the outputs repeat the same paragraph. All the output is a repetition of a meaningless response, showing no understanding of the task requirements. In contrast, models such as Llava-NeXT have fewer parameters but show an understanding of the task setup and give a response. This finding warrants subsequent exploration of the token level of generation.

Answer to RQ2-1: At the **Text and Image** level, LMMs can be experts on some specialized pre-trained tasks but are inferior to baseline methods for other tasks.

6.2.2 RQ2-2: At Text, Video level, do the LMMs have sufficient capabilities to help the multimodal software system development process and research?

Video input is a unique modality that differs significantly from simply using multiple images as input. In videos, there is a strong correlation and continuity between frames, requiring contextual understanding to interpret the content. For tasks involving this modality, we selected two tasks.

Video valid frame detection The first task is about detecting whether video frames are valid, as presented by Bao et al[6]. This task is a sub-task of a larger problem—extracting code from videos. Specifically, this task involves analyzing each frame of a video to determine whether it contains useful code content that needs to be extracted. If a frame contains such content, it is labeled as valid; otherwise, it is labeled as invalid.

It is important to note that this differs from analyzing a single image to determine its validity, as the validity of a video frame often depends on its context within the sequence, rather than solely on the content of the frame itself. As such, this task is well-suited for evaluating a model's capability in video understanding.

We used the video dataset provided by Bao et al. to conduct this evaluation[6]. Again, we use the data from the original paper as the baseline. Since current large models typically process video understanding by extracting frames and treating them as a set of images, we adopted the same frame extraction approach as Bao et al. to ensure experimental rigor. The evaluation metrics also follow the methodology presented in their paper. In this task, we use the label 1 to indicate that a frame is valid and the label 0 to indicate that a frame is invalid. The experiment result of this sub-task is shown in Table 13.

Table 13: Experiment Result of Video display detect [6]

Models	**Percision**	**Recall**	**F1**	**TP**	**FP**	**FN**
GPT-4o-2024-05-13	0.891	**0.891**	0.891	57	7	7
InternVL-8b	0.938	0.857	0.895	60	4	10
qwen-omni-turbo	**0.950**	0.860	**0.900**	61	3	10
Phi4-multimodal-instruct	0.891	0.851	0.870	57	7	10
Baseline	0.910	0.850	0.880	2459	256	445

It is important to note that, due to the uneven ditribution of positive and negative data, the result may overrating the real capbility of LMM.For example, although InternVL-8B appears to perform exceptionally well at first glance— even surpassing GPT4o in video understanding—this is actually due to an imbalance in the dataset, where the number of invalid frames is insufficient, resulting in too few negative samples. However, it is still undeniable that the multimodal large model performs very well on this task, requiring no additional training at all while maintaining a very high accuracy.

GUI Comprehension in video The second task evaluates LMM's ability to recognize graphical user interfaces (GUIs) in recorded videos, as proposed by Chen et al.[14] In this task, MLLMs attempt to understand GUI operations demonstrated in instructional videos and subsequently answer corresponding questions (e.g., "If the user wants to prioritize unread emails, which of the following actions should they take?"). The model selects the most appropriate answer from four given options.

The primary challenge lies in assessing LMMs' capability to both extract visual information from videos and perform logical reasoning based on the extracted information. Notably, this task goes beyond simple visual recognition by requiring models to determine appropriate subsequent actions based on observed operations.

Our evaluation adopts the dataset provided by chen et al.[14], using the baseline models presented in their original work. To ensure methodological consistency with previous tasks, we maintain identical frame extraction protocols. Unlike the original study's evaluation approach that employs separate scoring models, our measurement directly compares model outputs with ground-truth answers through exact match verification. This comprehensive dataset covers multiple operating systems and platforms including iOS, web interfaces, and extended reality (XR) systems, making it particularly suitable for demonstrating models' cross-platform understanding capabilities. The experimental results are detailed in Table 14.

Table 14: Experiment Result of GUI Comprehension of Video [14]

Models	Web	IOS	XR	Software
GPT-4o-2024-11-20	75%	83%	**84%**	**86%**
InternVL-8b	82%	75%	68%	79%
qwen-omni-turbo	**82%**	77%	78%	83%
Phi4-multimodal-instruct	80%	**85%**	81%	81%
Baseline	54%	51%	56%	60%

The aforementioned results demonstrate that current Large Multimodal Models (LMMs) exhibit remarkable proficiency in understanding dynamic GUI operations. They not only accurately interpret the content presented in videos but also logically infer correct answers by synthesizing contextual relationships between interface elements. Notably, while our prior experiment revealed LMMs' poor performance in precisely localizing GUI components within static screenshots, this limitation does not imply an inability to recognize these components. Significantly, even smaller-scale models like the 8B-parameter *InternVL* exhibit robust comprehension capabilities in this task.

These findings suggest that LMMs possess substantial potential for continued development in GUI understanding applications. Their demonstrated ability to analyze operational workflows and derive actionable insights positions them as promising tools to assist future GUI development processes, particularly in automating interface testing, enhancing user behavior analysis, and supporting adaptive interface design.

Answer to RQ2-2: At the text and video level, the LMMs have very strong potential to assist in this area, even achieving performance comparable to the baseline.

6.2.3 RQ2-3: At Text, Audio level, do the LMMs have sufficient capabilities to help the multimodal software system development process and research?

In order to serve the user like a Virtual personal assistant (VPA), an audio-capable LMM should be able to recognize the same meaning in different speech inputs, e.g., "What's up today?" and "Tell me the news of the day" should trigger the same news-playing state of a VPA. Based on the work of Guglielmi et al., we designed a series of tests to check whether the LMM is good at detecting the corresponding trigger state in the input text [36]. However, since the evaluation criterion used in the paper is to obtain the truth label through a textual conversation with the AWS skill VPA [10] in the simulator, in this experiment, we only tested the speech recognition accuracy of the LMM and then multiplied it by the original result that the LMM can test the accuracy and comprehensiveness of the VPA. For the sake

[10]https://explore.skillbuilder.aws/learn

of rigor, we only report the results of the Automatic Speech Recognition (ASR) part of the experiment in this report Table 15. We used SemaScore [79] as the evaluation metric of ASR accuracy, a criterion for determining the accuracy of ASR work from the language model token level.

Table 15: Experiment Result of Automatic Speech Recognition (ASR) [79]

Models	SemaScore
GPT-4o-audio-preview	**0.9583**
qwen-omni-turbo-2025-03-26	0.9433
Phi4-multimodal-instruct	0.4015

The experiment result indicates that LMM, like GPT-4o and Qwen-omni, has good speech recognition capabilities, and we believe that future LMMs can be used as VPAs to provide a broader range of services. However, the current support for speech input and the ability to analyze the non-textual information of speech are still lacking, and only some of the fine-tuned mini-models [115] [114] have good performance in this area. There is still a certain distance from the performance of solving text-level problems like LLM.

Another interesting finding here is the Phi4's performance. After checking the experiment result, we found that Phi4 did not follow the system and user prompt, which instructed the model to perform ASR tasks, but output what kind of sound label was inside the audio. For example, the audio saying "Alexa, open Smart Home", but Phi4 output "Labels: Human voice; Speech; Female speech and woman speaking", which is non-relevant to our prompts. To avoid the weak prompt issue, we re-do the same experiment with augmented prompting engineering. Nevertheless, Phi4 persists in outputting incorrect answers. This finding indicates that Phi4's prompt-following ability is a crucial problem for future usage.

Answer to RQ2-3: LMM can understand text information inside audio, so LMM has sufficient capabilities to help the multimodal software system development process and research.

In summary, large multimodal models have demonstrated exceptional capabilities in integrating with the field of software engineering across currently supported input modalities, including images, videos, and audio. For previously unseen tasks, these models can provide accurate and rapid responses based solely on prompts, often without requiring additional fine-tuning. Large multimodal models hold tremendous potential in the software engineering domain, offering researchers an incredibly practical tool for handling multimodal inputs without the need for extensive effort and time spent on fine-tuning or debugging.

Answer to RQ2: Large multimodal models exhibit exceptional performance within currently supported modalities, highlighting significant potential for integration into the field of software engineering.

7 Conclusion

The rapid advancement of multimodal large language models (LMMs) has shifted research focus toward their practical applications – *what can we achieve with LMMs?* However, our systematic survey reveals critical gaps in software engineering research: while numerous studies attempt to apply MLLMs to domain-specific problems, there exists no comprehensive benchmarking framework to systematically evaluate model capabilities, quantify performance limitations, and guide future research directions. Two primary challenges emerge from our analysis:

- **Lack of unified evaluation**: Current efforts remain fragmented across subdomains without standardized metrics or comparative baselines
- **Task collection barrier**: Relevant evaluation tasks are dispersed across disparate research fields, creating significant overhead for researchers

In conclusion, the following are some of the main contributions we have made during this project.

- **Taxonomy**: We undertook a comprehensive systematic literature review to establish a conceptual taxonomy delineating research domains where scholarly publications (PAPERS) can derive substantive benefits from Large Multimodal Models (LMMs). Building upon this foundation, we implemented a dual-phase analytical framework employing both conventional Large Language Models (LLMs) and specialized Reasoning-Enhanced LLMs to systematically categorize and synthesize an extensive corpus of publications through our taxonomy.

- **Empirical analysis**: We present a structured analysis of methodological implications through a task-oriented hierarchy derived from our taxonomy. This analytical framework provides new insights into how multimodal approaches can advance software system research, particularly in addressing complex integration challenges across heterogeneous data modalities.
- **Testing framework**: We developed and validated a modular evaluation framework with three core innovations: (1) Unified benchmarking infrastructure supporting concurrent assessment of heterogeneous models, datasets, and metrics; (2) Extensible architecture facilitating seamless integration of novel test components; (3) Multi-dimensional evaluation protocol combining technical performance metrics with practical utility analysis.
- **Experiment**: Through rigorous experimentation on state-of-the-art LMMs, we executed a curated set of benchmark tasks to quantitatively assess model capabilities while simultaneously evaluating their effectiveness in real-world developer-assistance scenarios.

References

[1] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report, 2024. 49

[2] Mohammad D Alahmadi. Vid2xml: Automatic extraction of a complete xml data from mobile programming screencasts. *IEEE Transactions on Software Engineering*, 49(4):1726–1740, 2022. 52

[3] Luca Ardito, Andrea Bottino, Riccardo Coppola, Fabrizio Lamberti, Francesco Manigrasso, Lia Morra, and Marco Torchiano. Feature matching-based approaches to improve the robustness of android visual gui testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(2):1–32, 2021. 51

[4] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*, 2021. 51

[5] Haoli Bai, Zhiguang Liu, Xiaojun Meng, Wentao Li, Shuang Liu, Nian Xie, Rongfu Zheng, Liangwei Wang, Lu Hou, Jiansheng Wei,

et al. Wukong-reader: Multi-modal pre-training for fine-grained visual document understanding. *arXiv preprint arXiv:2212.09621*, 2022. 51

[6] Lingfeng Bao, Zhenchang Xing, Xin Xia, David Lo, Minghui Wu, and Xiaohu Yang. psc2code: Denoising code extraction from programming screencasts. *ACM Transactions on Software Engineering and Methodology*, 29(3):1–38, June 2020. 51, 52, 53, 71, 72

[7] Arnab Barua, Mobyen Uddin Ahmed, and Shahina Begum. A systematic literature review on multimodal machine learning: Applications, challenges, gaps and future directions. *IEEE Access*, 11:14804–14831, 2023. 12

[8] Carlos Bernal-Cárdenas, Nathan Cooper, Madeleine Havranek, Kevin Moran, Oscar Chaparro, Denys Poshyvanyk, and Andrian Marcus. Translating video recordings of complex mobile app ui gestures into replayable scenarios. *IEEE Transactions on Software Engineering*, 49(4):1782–1803, 2022. 9, 51

[9] Rizhao Cai, Zirui Song, Dayan Guan, Zhenhao Chen, Yaohang Li, Xing Luo, Chenyu Yi, and Alex Kot. Benchlmm: Benchmarking cross-style visual capability of large multimodal models. In *European Conference on Computer Vision*, pages 340–358. Springer, 2025. 13

[10] Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Yuchen Mao,

Wenjing Hu, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. Spider2-v: How far are multimodal agents from automating data science and engineering workflows?, 2024. 13

[11] Liwei Chan, Yi-Chi Liao, George B Mo, John J Dudley, Chun-Lien Cheng, Per Ola Kristensson, and Antti Oulasvirta. Investigating positive and negative qualities of human-in-the-loop optimization for designing interaction techniques. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2022. 42

[12] Prantik Chatterjee, José Campos, Rui Abreu, and Subhajit Roy. Augmenting automated spectrum based fault localization for multiple faults. In *IJCAI*, pages 3140–3148, 2023. 51

[13] Anthony Chen, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. Purr: Efficiently editing language model hallucinations by denoising language model corruptions, 2023. 44

[14] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding, 2025. 51, 52, 53, 73, 74

[15] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, pages 322–334, 2020. 52

[16] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Zhu Liming Xu, Xiwei, and Guoqiang Li. Object detection for graphical user interface: Old fashioned or deep learning or a combination? In *Proceedings of the 2020 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, 2020. ACM. 52, 53, 67

[17] Ke Chen, Yufei Li, Yingfeng Chen, Changjie Fan, Zhipeng Hu, and Wei Yang. Glib: towards automated test oracle for graphically-rich applications. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1093–1104, 2021. 52

[18] Liguo Chen, Qi Guo, Hongrui Jia, Zhengran Zeng, Xin Wang, Yijiang Xu, Jian Wu, Yidong Wang, Qing Gao, Jindong Wang, Wei Ye, and Shikun Zhang. A survey on evaluating large language models in code generation tasks, 2025. 39, 40

[19] Liuqing Chen, Yunnong Chen, Shuhong Xiao, Yaxuan Song, Lingyun Sun, Yankun Zhen, Tingting Zhou, and Yanfang Chang. Egfe: End-to-end grouping of fragmented elements in ui designs with multimodal learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–12, 2024. 41, 52

[20] Taizhou Chen, Lantian Xu, Xianshan Xu, and Kening Zhu. Gestonhmd: Enabling gesture-based interaction on low-cost vr head-mounted display. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2597–2607, 2021. 51

[21] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling, 2025. 49

[22] Yan Chen, Maulishree Pandey, Jean Y Song, Walter S Lasecki, and Steve Oney. Improving crowd-supported gui testing with structural guidance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020. 51

[23] Yinpeng Chen, Zhiyu Pan, Min Shi, Hao Lu, Zhiguo Cao, and Weicai Zhong. Design what you desire: Icon generation from orthogonal application and theme labels. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2536–2546, 2022. 51

[24] Zimin Chen, Vincent J Hellendoorn, Pascal Lamblin, Petros Maniatis, Pierre-Antoine Manzagol, Daniel Tarlow, and Subhodeep Moitra. Plur: A unifying, graph-based view of program learning, understanding, and repair. *Advances in Neural Information Processing Systems*, 34:23089–23101, 2021. 51

[25] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. 52, 67, 68

[26] Shi Cheng, Xueping Wang, Mingming Zhang, Xiujuan Lei, Hui Lu, and Yuhui Shi. Solving multimodal optimization problems by a knowledge-driven brain storm optimization algorithm. *Applied Soft Computing*, 150:111105, 2024. 42

[27] K Dilşad Çiçek, Taymaz Akan, and Oğuz Bayat. Multi-modal battle royale optimizer. *Cluster Computing*, 27(7):8983–8993, 2024. 42

[28] John W Creswell and Cheryl N Poth. *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications, 2016. 15

[29] Zhenlong Dai, Bingrui Chen, Zhuoluo Zhao, Xiu Tang, Sai Wu, Chang Yao, Zhipeng Gao, and Jingyuan Chen. Less is more: Adaptive program repair with bug localization and preference learning, 2025. 39

[30] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin

Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. 22, 24

[31] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854, 2017. 51

[32] Feng Dong, Haoyu Wang, Li Li, Yao Guo, Tegawendé F Bissyandé,

Tianming Liu, Guoai Xu, and Jacques Klein. Frauddroid: Automated ad fraud detection for android apps. In *Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, pages 257–268, 2018. 51

[33] Sevinc Eroglu, Arthur Voigt, Benjamin Weyers, and Torsten W Kuhlen. Vrscenariobuilder: Free-hand immersive authoring tool for scenario-based testing of automated vehicles. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 196–202. IEEE, 2024. 51

[34] Ankita Gandhi, Kinjal Adhvaryu, Soujanya Poria, Erik Cambria, and Amir Hussain. Multimodal sentiment analysis: A systematic review of history, datasets, multimodal fusion methods, applications, challenges and future directions. *Information Fusion*, 91:424–444, 2023. 11

[35] Tianxiao Gu, Chengnian Sun, Xiaoxing Ma, Chun Cao, Chang Xu, Yuan Yao, Qirun Zhang, Jian Lu, and Zhendong Su. Practical gui testing of android applications via model abstraction and refinement. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 269–280. IEEE, 2019. 51

[36] Emanuela Guglielmi, Giovanni Rosa, Simone Scalabrino, Gabriele Bavota, and Rocco Oliveto. Help them understand: Testing and

improving voice user interfaces. *ACM Transactions on Software Engineering and Methodology*, 33(6):1–33, 2024. 11, 75

[37] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.*, September 2024. Just Accepted. 12

[38] Jiajun Hu, Lili Wei, Yepang Liu, Shing-Chi Cheung, and Huaxun Huang. A tale of two cities: How webview induces bugs to android applications. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 702–713, 2018. 51

[39] Pascal Jansen, Julian Britten, Alexander Häusele, Thilo Segschneider, Mark Colley, and Enrico Rukzio. Autovis: Enabling mixed-immersive analysis of automotive user interface interaction studies. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2023. 51

[40] Juan José Montero Jimenez, Sébastien Schwartz, Rob Vingerhoeds, Bernard Grabot, and Michel Salaün. Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of manufacturing systems*, 56:539–557, 2020. 40

[41] Haolin Jin, Linghan Huang, Haipeng Cai, Jun Yan, Bo Li, and Huaming Chen. From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479*, 2024. 12

[42] Harshit Joshi, José Cambronero Sanchez, Sumit Gulwani, Vu Le, Gust Verbruggen, and Ivan Radiček. Repair is nearly generation: Multilingual program repair with llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5131–5140, 2023. 51

[43] Naoki Kimura, Tan Gemicioglu, Jonathan Womack, Richard Li, Yuhui Zhao, Abdelkareem Bedri, Zixiong Su, Alex Olwal, Jun Rekimoto, and Thad Starner. Silentspeller: Towards mobile, hands-free, silent speech text entry using electropalatography. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2022. 51

[44] Yuanhong Lan, Yifei Lu, Zhong Li, Minxue Pan, Wenhua Yang, Tian Zhang, and Xuandong Li. Deeply reinforcing android gui testing with deep reinforcement learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024. 51

[45] Ricardo Langner, Marc Satkowski, Wolfgang Büschel, and Raimund Dachselt. Marvis: Combining mobile devices and augmented reality for visual data analysis. In *Proceedings of the 2021*

CHI Conference on Human Factors in Computing Systems, pages 1–17, 2021. 51

[46] Gaotang Li, Jiarui Liu, and Wei Hu. Bias amplification enhances minority group performance. *arXiv preprint arXiv:2309.06717*, 2023. 45

[47] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and ZHAO-XIANG ZHANG. Sheetcopilot: Bringing software productivity to the next level through large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 51

[48] Kaixin Li, Meng ziyang, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: GUI grounding for professional high-resolution computer use. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. 52, 67, 70

[49] Linlin Li, Ruifeng Wang, Xian Zhan, Ying Wang, Cuiyun Gao, Sinan Wang, and Yepang Liu. What you see is what you get? it is not the case! detecting misleading icons for mobile applications. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 538–550, 2023. 52

[50] Suwan Li, Lei Bu, Guangdong Bai, Zhixiu Guo, Kai Chen, and Hanlin Wei. Vitas: Guided model-based vui testing of vpa apps. In *Proceedings of the 37th IEEE/ACM International Conference on Auto-*

mated Software Engineering, ASE '22, New York, NY, USA, 2023. Association for Computing Machinery. 52

[51] Xingjun Li, Yuanxin Wang, Hong Wang, Yang Wang, and Jian Zhao. Nbsearch: Semantic search and visual exploration of computational notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2021. 51

[52] Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioner for conjugate gradient pde solvers, 2023. 45

[53] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024. 12

[54] Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, and Zhaoxuan Chen. Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2244–2258, 2021. 51

[55] Paul Pu Liang, Yiwei Lyu, Gunjan Chhablani, Nihal Jain, Zihao Deng, Xingbo Wang, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multiviz: Towards user-centric visualizations and interpretations of multimodal models. In *Extended Abstracts of the*

2023 CHI Conference on Human Factors in Computing Systems, pages 1–21, 2023. 51

[56] Jun-Wei Lin, Navid Salehnamadi, and Sam Malek. Route: Roads not taken in ui testing. *ACM Transactions on Software Engineering and Methodology*, 32(3):1–25, 2023. 51

[57] Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, Chris White, Quanquan Gu, Jian Pei, Carl Yang, and Liang Zhao. Domain specialization as the key to make large language models disruptive: A comprehensive survey, 2024. 42

[58] Qihao Liu, Adam Kortylewski, and Alan L Yuille. Poseexaminer: Automated testing of out-of-distribution robustness in human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 672–681, 2023. 51

[59] Qin Liu, Chao Shang, Ling Liu, Nikolaos Pappas, Jie Ma, Neha Anna John, Srikanth Doss, Lluis Marquez, Miguel Ballesteros, and Yassine Benajiba. Unraveling and mitigating safety alignment degradation of vision-language models, 2024. 44

[60] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. Owl eyes: spotting ui display issues via visual un-

derstanding. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, page 398–409. ACM, December 2020. 9, 52, 53, 64, 65

[61] Zhijie Liu, Liang Feng Zhang, and Yutian Tang. Enhancing malware detection for android apps: Detecting fine-granularity malicious components. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1212–1224. IEEE, 2023. 42, 52

[62] Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On llms-driven synthetic data generation, curation, and evaluation: A survey, 2024. 12

[63] M Ronnier Luo, Guihua Cui, and Bryan Rigg. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 26(5):340–350, 2001. 62

[64] Finlay Macklon, Mohammad Reza Taesiri, Markos Viggiato, Stefan Antoszko, Natalia Romanova, Dale Paas, and Cor-Paul Bezemer. Automatically detecting visual bugs in html5 canvas games. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–11, 2022. 52

[65] Finlay Macklon, Markos Viggiato, Natalia Romanova, Chris Buzon, Dale Paas, and Cor-Paul Bezemer. A taxonomy of testable html5 canvas issues. *IEEE Transactions on Software Engineering*, 49(6):3647–3659, 2023. 52

[66] Vikas K Malviya, Yan Naing Tun, Chee Wei Leow, Ailys Tee Xynyn, Lwin Khin Shar, and Lingxiao Jiang. Fine-grained in-context permission classification for android apps using control-flow graph embedding. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1225–1237. IEEE, 2023. 51

[67] SM Hasan Mansur, Sabiha Salma, Damilola Awofisayo, and Kevin Moran. Aidui: Toward automated recognition of dark patterns in user interfaces. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1958–1970. IEEE, 2023. 42, 52

[68] Leonardo Mariani, Ali Mohebbi, Mauro Pezzè, and Valerio Terragni. Semantic matching of gui events for test reuse: are we there yet? In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 177–190, 2021. 51

[69] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023. 20

[70] Waterfall Model. Waterfall model. *Luettavissa: http://www. waterfall-model. com/. Luettu*, 3, 2015. 15

[71] Mohammad Amin Mozaffari, Xinyuan Zhang, Jinghui Cheng, and Jin LC Guo. Ganspiration: Balancing targeted and serendipitous inspiration in user interface design with style-based generative adversarial network. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2022. 51

[72] Thao Nguyen, Gabriel Ilharco, Mitchell Wortsman, Sewoong Oh, and Ludwig Schmidt. Quality not quantity: On the interaction between dataset design and robustness of clip, 2023. 39

[73] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas

Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakan-tan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Gi-ambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Rad-ford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ry-der, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sas-try, Heather Schmidt, David Schnurr, John Schulman, Daniel Sel-sam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jor-dan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thomp-son, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tug-gle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Ja-

son Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. 9

[74] Chae-Yeun Park and Michael J. Kastoryano. Complexity phase transitions in instantaneous quantum polynomial-time circuits, 2023. 43

[75] Sara Pieri, Jose Restom, Samuel Horvath, and Hisham Cholakkal. Handling data heterogeneity via architectural design for federated visual recognition. *Advances in Neural Information Processing Systems*, 36:4115–4136, 2023. 51

[76] Jing Qi, Li Ma, Zhenchao Cui, and Yushu Yu. Computer vision-based hand gesture recognition for human-robot interaction: a review. *Complex & Intelligent Systems*, 10(1):1581–1606, 2024. 11

[77] Patrik Reizinger, Szilvia Ujváry, Anna Mészáros, Anna Kerekes, Wieland Brendel, and Ferenc Huszár. Position: Understanding llms requires more than statistical generalization, 2024. 11

[78] Navid Salehnamadi, Ziyao He, and Sam Malek. Assistive-technology aided manual accessibility testing in mobile apps, powered by record-and-replay. In *Proceedings of the 2023 CHI Confer-*

ence on Human Factors in Computing Systems, pages 1–20, 2023. 40

[79] Zitha Sasindran, Harsha Yelchuri, and T. V. Prabhakar. Semascore: A new evaluation metric for automatic speech recognition tasks. In *Interspeech 2024*, interspeech$_2024, page4558–4562. ISCA, September$2024. 53, 76

[80] JooYoung Seo, Yilin Xia, Bongshin Lee, Sean McCurry, and Yu Jun Yam. Maidr: Making statistical visualizations accessible with multimodal data representation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2024. 51

[81] Fei Shao, Rui Xu, Wasif Haque, Jingwei Xu, Ying Zhang, Wei Yang, Yanfang Ye, and Xusheng Xiao. Webevo: taming web application evolution via detecting semantic structure changes. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 16–28, 2021. 52

[82] Chenglei Si, Yanzhe Zhang, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: How far are we from automating front-end engineering?, 2024. 52, 53, 62, 63

[83] Luis R Soenksen, Yu Ma, Cynthia Zeng, Leonard Boussioux, Kimberly Villalobos Carballo, Liangyuan Na, Holly M Wiberg, Michael L Li, Ignacio Fuentes, and Dimitris Bertsimas. Integrated multimodal artificial intelligence framework for healthcare applications. *NPJ digital medicine*, 5(1):149, 2022. 39, 45

[84] Ting Su, Jue Wang, and Zhendong Su. Benchmarking automated gui testing for android against real-world bugs. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 119–130, 2021. 51

[85] Yuhui Su, Chunyang Chen, Junjie Wang, Zhe Liu, Dandan Wang, Shoubin Li, and Qing Wang. The metamorphosis: Automatic detection of scaling issues for mobile apps. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022. 52

[86] Cong Sun, Yuwan Ma, Dongrui Zeng, Gang Tan, Siqi Ma, and Yafei Wu. μdep: Mutation-based dependency generation for precise taint analysis on android native code. *IEEE Transactions on Dependable and Secure Computing*, 20(2):1461–1475, 2022. 51

[87] Qin Sun, Yunqi Hu, Mingming Fan, Jingting Li, and Su-Jing Wang. “can it be customized according to my motor abilities?”: Toward designing user-defined head gestures for people with dystonia. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2024. 51

[88] Mohammad Reza Taesiri, Tianjun Feng, Cor-Paul Bezemer, and Anh Nguyen. Glitchbench: Can large multimodal models detect video game glitches? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22444–22455, 2024. 52

[89] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*, 2023. 12

[90] Ruixiang Tang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. Does synthetic data generation of llms help clinical text mining?, 2023. 12

[91] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Jack Krawczyk, Cosmo Du, Ed Chi, Heng-Tze Cheng, Eric Ni, Purvi Shah, Patrick Kane, Betty Chan, Manaal Faruqui, Aliaksei Severyn, Hanzhao Lin, YaGuang Li, Yong Cheng, Abe Ittycheriah, Mahdis Mahdieh, Mia Chen, Pei Sun, Dustin Tran, Sumit Bagri, Balaji Lakshminarayanan, Jeremiah Liu, Andras Orban, Fabian Güra, Hao Zhou, Xinying Song, Aurelien Boffy, Harish Ganapathy, Steven Zheng, HyunJeong Choe, Ágoston Weisz, Tao Zhu, Yifeng Lu, Siddharth Gopal, Jarrod Kahn, Maciej Kula, Jeff Pitman, Rushin Shah, Emanuel Taropa, Majd Al Merey, Martin Baeuml, Zhifeng Chen, Laurent El Shafey, Yujing Zhang, Olcan Sercinoglu, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka,

Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Gaurav Singh Tomar, Evan Senter, Martin Chadwick, Ilya Kornakov, Nithya Attaluri, Iñaki Iturrate, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Xavier Garcia, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Ravi Addanki, Antoine Miech, Annie Louis, Denis Teplyashin, Geoff Brown, Elliot Catt, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaliy Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan

Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Sidharth Mudgal, Romina Stella, Kevin Brooks, Gautam Vasudevan, Chenxi Liu, Mainak Chain, Nivedita Melinkeri, Aaron Cohen, Venus Wang, Kristie Seymore, Sergey Zubkov, Rahul Goel, Summer Yue, Sai Krishnakumaran, Brian Albert, Nate Hurley, Motoki Sano, Anhad Mohananey, Jonah Joughin, Egor Filonov, Tomasz Kępa, Yomna Eldawy, Jiawern Lim, Rahul Rishi, Shirin Badiezadegan, Taylor Bos, Jerry

Chang, Sanil Jain, Sri Gayatri Sundara Padmanabhan, Subha Puttagunta, Kalpesh Krishna, Leslie Baker, Norbert Kalb, Vamsi Bedapudi, Adam Kurzrok, Shuntong Lei, Anthony Yu, Oren Litvin, Xiang Zhou, Zhichun Wu, Sam Sobell, Andrea Siciliano, Alan Papir, Robby Neale, Jonas Bragagnolo, Tej Toor, Tina Chen, Valentin Anklin, Feiran Wang, Richie Feng, Milad Gholami, Kevin Ling, Lijuan Liu, Jules Walter, Hamid Moghaddam, Arun Kishore, Jakub Adamek, Tyler Mercado, Jonathan Mallinson, Siddhinita Wandekar, Stephen Cagle, Eran Ofek, Guillermo Garrido, Clemens Lombriser, Maksim Mukha, Botu Sun, Hafeezul Rahman Mohammad, Josip Matak, Yadi Qian, Vikas Peswani, Pawel Janus, Quan Yuan, Leif Schelin, Oana David, Ankur Garg, Yifan He, Oleksii Duzhyi, Anton Älgmyr, Timothée Lottaz, Qi Li, Vikas Yadav, Luyao Xu, Alex Chinien, Rakesh Shivanna, Aleksandr Chuklin, Josie Li, Carrie Spadine, Travis Wolfe, Kareem Mohamed, Subhabrata Das, Zihang Dai, Kyle He, Daniel von Dincklage, Shyam Upadhyay, Akanksha Maurya, Luyan Chi, Sebastian Krause, Khalid Salama, Pam G Rabinovitch, Pavan Kumar Reddy M, Aarush Selvan, Mikhail Dektiarev, Golnaz Ghiasi, Erdem Guven, Himanshu Gupta, Boyi Liu, Deepak Sharma, Idan Heimlich Shtacher, Shachi Paul, Oscar Akerlund, François-Xavier Aubet, Terry Huang, Chen Zhu, Eric Zhu, Elico Teixeira, Matthew Fritze, Francesco Bertolini, Liana-Eleonora Marinescu, Martin Bölle, Dominik Paulus, Khyatti Gupta, Tejasi Latkar, Max Chang, Jason Sanders, Roopa Wilson, Xuewei Wu, Yi-Xuan Tan, Lam Nguyen Thiet, Tulsee Doshi, Sid Lall, Swaroop Mishra, Wanming Chen, Thang Luong, Seth Benjamin, Jasmine Lee, Ewa Andrejczuk, Dominik Rabiej, Vipul Ranjan, Krzysztof Styrc, Pengcheng Yin, Jon Simon, Malcolm Rose Harriott, Mudit Bansal, Alexei Robsky, Geoff Bacon, David Greene, Daniil Mirylenka, Chen Zhou, Obaid Sarvana, Abhimanyu Goyal, Samuel Andermatt, Patrick Siegler, Ben Horn, Assaf Israel, Francesco Pongetti, Chih-Wei ”Louis” Chen, Marco Selvatici, Pedro Silva, Kathie Wang, Jackson Tolins, Kelvin Guu, Roey Yogev, Xiaochen Cai, Alessandro Agostini, Maulik Shah, Hung Nguyen, Noah Ó Donnaile, Sébastien Pereira, Linda Friso, Adam Stambler, Adam Kurzrok, Chenkai Kuang, Yan Romanikhin, Mark Geller, ZJ Yan, Kane Jang, Cheng-Chun Lee, Wojciech Fica, Eric Malmi, Qijun Tan, Dan Banica, Daniel Balle, Ryan Pham, Yanping Huang, Diana Avram, Hongzhi Shi, Jasjot Singh, Chris Hidey, Niharika Ahuja, Pranab Saxena, Dan Dooley, Srividya Pranavi Potharaju, Eileen O’Neill, Anand Gokulchandran, Ryan Foley, Kai Zhao, Mike Dusenberry, Yuan Liu, Pulkit Mehta, Ragha Kotikalapudi, Chalence Safranek-Shrader, Andrew Goodman, Joshua Kessinger, Eran Globen, Prateek Kolhar, Chris Gorgolewski, Ali Ibrahim, Yang Song, Ali Eichenbaum, Thomas Brovelli, Sahitya Potluri, Preethi Lahoti, Cip Baetu, Ali Ghorbani, Charles Chen, Andy Crawford, Shalini Pal, Mukund Sridhar, Petru Gurita, Asier Mujika, Igor Petrovski, Pierre-Louis Cedoz, Chenmei Li, Shiyuan Chen, Niccolò Dal Santo, Siddharth Goyal, Jitesh Punjabi, Karthik Kappaganthu, Chester Kwak, Pallavi LV, Sarmishta Velury, Himadri Choudhury, Jamie Hall, Premal Shah, Ricardo Figueira, Matt Thomas, Minjie Lu, Ting Zhou, Chintu Kumar, Thomas Jurdi, Sharat Chikkerur, Yenai Ma, Adams Yu, Soo Kwak, Victor Ähdel, Sujeevan Rajayogam, Travis Choma, Fei Liu, Aditya Barua, Colin Ji, Ji Ho Park, Vincent Hellendoorn, Alex Bailey, Taylan Bilal, Huanjie Zhou, Mehrdad Khatir, Charles Sutton, Wojciech Rzadkowski, Fiona Macintosh, Konstantin Shagin, Paul Medina, Chen Liang, Jinjing Zhou, Pararth Shah, Yingying Bi, Attila Dankovics, Shipra Banga, Sabine Lehmann, Marissa Bredesen, Zifan Lin, John Eric Hoffmann, Jonathan Lai, Raynald Chung, Kai Yang, Nihal Balani, Arthur Bražinskas, Andrei Sozanschi, Matthew Hayes, Héctor Fernández Alcalde, Peter Makarov, Will Chen, Antonio Stella, Liselotte Snijders, Michael Mandl, Ante Kärrman, Paweł Nowak, Xinyi Wu, Alex Dyck, Krishnan Vaidyanathan, Raghavender R, Jessica Mallet, Mitch Rudominer, Eric Johnston, Sushil Mittal, Akhil Udathu, Janara Christensen, Vishal Verma, Zach Irving, Andreas Santucci, Gamaleldin Elsayed, Elnaz Davoodi, Marin Georgiev, Ian Tenney, Nan Hua, Geoffrey Cideron, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Dylan Scandinaro, Heinrich Jiang, Jasper Snoek, Mukund Sundararajan, Xuezhi Wang, Zack Ontiveros, Itay Karo, Jeremy Cole, Vinu Rajashekhar, Lara Tumeh, Eyal Ben-David, Rishub Jain, Jonathan Uesato, Romina Datta, Oskar Bunyan, Shimu Wu, John Zhang, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Jane Park, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Geoffrey Irving, Edward

Loper, Michael Fink, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Ivan Petrychenko, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Evan Palmer, Paul Suganthan, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Ginger Perng, Elena Allica Abellan, Mingyang Zhang, Ishita Dasgupta, Nate Kushman, Ivo Penchev, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Daniel Andor, Pedro Valenzuela, Minnie Lui, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Ken Franko, Anna Bulanova, Rémi Leblond, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Mark Omernick, Colton Bishop, Rachel Sterneck, Rohan Jain, Jiawei Xia, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Daniel J. Mankowitz, Alex Polozov, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Matthieu Geist, Ser tan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Kathy Wu, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Saaber Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Yeqing Li, Nir Levine, Ariel Stolovich, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Charlie Deck, Hyo Lee, Zonglin Li, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Sho Arora, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart

Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Lynette Webb, Sahil Dua, Dong Li, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Evgenii Eltyshev, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng

Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Christof Angermueller, Xiaowei Li, Anoop Sinha, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Denny Zhou, Komal Jalan, Dinghua Li, Blake Hechtman, Parker Schuh, Milad Nasr, Kieran Milan, Vladimir Mikulik, Juliana Franco, Tim Green, Nam Nguyen, Joe Kelley, Aroma Mahendru, Andrea Hu, Joshua Howland, Ben Vargas, Jeffrey Hui, Kshitij Bansal, Vikram Rao, Rakesh Ghiya, Emma Wang, Ke Ye, Jean Michel Sarr, Melanie Moranski Preston, Madeleine Elish, Steve Li, Aakash Kaku, Jigar Gupta, Ice Pasupat, Da-Cheng Juan, Milan Someswar, Tejvi M., Xinyun Chen, Aida Amini, Alex Fabrikant, Eric Chu, Xuanyi Dong, Amruta Muthal, Senaka Buthpitiya, Sarthak Jauhari, Nan Hua, Urvashi Khandelwal, Ayal Hitron, Jie Ren, Larissa Rinaldi, Shahar Drath, Avigail Dabush, Nan-Jiang Jiang, Harshal Godhia, Uli Sachs, Anthony Chen, Yicheng Fan, Hagai Taitelbaum, Hila Noga, Zhuyun Dai, James Wang, Chen Liang, Jenny Hamer, Chun-Sung Ferng, Chenel Elkind, Aviel Atias, Paulina Lee, Vít Listík, Mathias Carlen, Jan van de Kerkhof, Marcin Pikus, Krunoslav Zaher, Paul Müller, Sasha Zykova, Richard Stefanec, Vitaly Gatsko, Christoph Hirnschall, Ashwin Sethi, Xingyu Federico Xu, Chetan Ahuja, Beth Tsai, Anca Stefanoiu, Bo Feng, Keshav

Dhandhania, Manish Katyal, Akshay Gupta, Atharva Parulekar, Divya Pitta, Jing Zhao, Vivaan Bhatia, Yashodha Bhavnani, Omar Alhadlaq, Xiaolin Li, Peter Danenberg, Dennis Tu, Alex Pine, Vera Filippova, Abhipso Ghosh, Ben Limonchik, Bhargava Urala, Chaitanya Krishna Lanka, Derik Clive, Yi Sun, Edward Li, Hao Wu, Kevin Hongtongsak, Ianna Li, Kalind Thakkar, Kuanysh Omarov, Kushal Majmundar, Michael Alverson, Michael Kucharski, Mohak Patel, Mudit Jain, Maksim Zabelin, Paolo Pelagatti, Rohan Kohli, Saurabh Kumar, Joseph Kim, Swetha Sankar, Vineet Shah, Lakshmi Ramachandruni, Xiangkai Zeng, Ben Bariach, Laura Weidinger, Tu Vu, Alek Andreev, Antoine He, Kevin Hui, Sheleem Kashem, Amar Subramanya, Sissie Hsiao, Demis Hassabis, Koray Kavukcuoglu, Adam Sadovsky, Quoc Le, Trevor Strohman, Yonghui Wu, Slav Petrov, Jeffrey Dean, and Oriol Vinyals. Gemini: A family of highly capable multimodal models, 2024. 9

[92] Johannes Treutlein, Dami Choi, Jan Betley, Samuel Marks, Cem Anil, Roger Grosse, and Owain Evans. Connecting the dots: Llms can infer and verbalize latent structure from disparate training data. *arXiv preprint arXiv:2406.14546*, 2024. 20

[93] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510, 2021. 38, 51

[94] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang

Lu, Yiqiang Chen, Wenjun Zeng, and S Yu Philip. Generalizing to unseen domains: A survey on domain generalization. *IEEE transactions on knowledge and data engineering*, 35(8):8052–8072, 2022. 11

[95] Jue Wang, Yanyan Jiang, Chang Xu, Chun Cao, Xiaoxing Ma, and Jian Lu. Combodroid: generating high-quality test inputs for android apps via use case combinations. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 469–480, 2020. 51

[96] Liping Wang, Jiawei Li, Lifan Zhao, Zhizhuo Kou, Xiaohan Wang, Xinyi Zhu, Hao Wang, Yanyan Shen, and Lei Chen. Methods for acquiring and incorporating knowledge into stock price prediction: A survey, 2023. 43

[97] Jeremy Warner. Visual design reuse through style recognition and transfer. In *Adjunct Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology*, pages 175–178, 2021. 38

[98] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. 13, 53

[99] Chaoyi Wu, Jiayu Lei, Qiaoyu Zheng, Weike Zhao, Weixiong Lin, Xiaoman Zhang, Xiao Zhou, Ziheng Zhao, Ya Zhang, Yanfeng Wang, et al. Can gpt-4v (ision) serve medical applications? case studies on gpt-4v for multimodal medical diagnosis. *arXiv preprint arXiv:2310.09909*, 2023. 13

[100] Guande Wu, Chen Zhao, Claudio Silva, and He He. Your co-workers matter: Evaluating collaborative capabilities of language models in blocks

world, 2024. 43

[101] Hao Lin Zhenhua Li Feng Qian Xianlong Wang Yunhao Liu Xinlei Yang, Wei Liu and Tianyin Xu. Visual-aware testing and debugging for web performance optimization. In *Proceedings of the International World Wide Web Conference (WWW)*. ACM, 2023. 51

[102] Bo Yang, Zhenchang Xing, Xin Xia, Chunyang Chen, Deheng Ye, and Shanping Li. Don't do that! hunting down visual design smells in complex uis against design guidelines. In *2021 IEEE/ACM 43rd international conference on software engineering (ICSE)*, pages 761–772. IEEE, 2021. 41

[103] Wangrui Yang et al. Legal regulation of intellectual property rights in the digital age: A perspective from aigc infringement. *Science of Law Journal*, 3(3):164–173, 2024. 45

[104] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, page nwae403, 2024. 12

[105] Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. Mm-llms: Recent advances in multimodal large language models, 2024. 12

[106] Guanhua Zhang, Yihua Zhang, Yang Zhang, Wenqi Fan, Qing Li, Sijia Liu, and Shiyu Chang. Fairness reprogramming. In Alice H. Oh, Alekh

Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 45

[107] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. Jellyfish: A large language model for data preprocessing. *arXiv preprint arXiv:2312.01678*, 2023. 12

[108] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. Large language models as data preprocessors. *arXiv preprint arXiv:2308.16361*, 2023. 12

[109] Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Reality check on the evaluation of large multimodal models, 2024. 44

[110] Zhaoxu Zhang, Robert Winn, Yu Zhao, Tingting Yu, and William GJ Halfond. Automatically reproducing android bug reports using natural language processing and reinforcement learning. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 411–422, 2023. 38, 41

[111] Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang. Seenomaly: Vision-based linting of gui animation effects against design-don't guidelines. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, pages 1286–1297, 2020. 51

[112] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey, 2023. 44

[113] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. Guigan: Learning to generate gui designs using generative adversarial networks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 748–760. IEEE, 2021. 41, 51

[114] Yang Zhao, Zhijie Lin, Daquan Zhou, Zilong Huang, Jiashi Feng, and Bingyi Kang. Bubogpt: Enabling visual grounding in multi-modal llms, 2023. 76

[115] Zhisheng Zheng, Puyuan Peng, Ziyang Ma, Xie Chen, Eunsol Choi, and David Harwath. Bat: Learning to reason about spatial sounds with large language models. *arXiv preprint arXiv:2402.01591*, 2024. 76