TOWARDS RESPONSIBLE AI IN THE ERA OF CHATGPT: A REFERENCE ARCHITECTURE FOR DESIGNING FOUNDATION MODEL-BASED AI SYSTEMS

Qinghua Lu[*], Liming Zhu, Xiwei Xu, Zhenchang Xing, Jon Whittle
Data61, CSIRO, Australia

May 24, 2023

ABSTRACT

The release of ChatGPT, Bard, and other large language model (LLM)-based chatbots has drawn huge attention on foundations models worldwide. There is a growing trend that foundation models will serve as the fundamental building blocks for most of the future AI systems. However, incorporating foundation models in AI systems raises significant concerns about responsible AI due to their black box nature and rapidly advancing super-intelligence. Additionally, the foundation model's growing capabilities can eventually absorb the other components of AI systems, introducing the moving boundary and interface evolution challenges in architecture design. To address these challenges, this paper proposes a pattern-oriented responsible-AI-by-design reference architecture for designing foundation model-based AI systems. Specially, the paper first presents an architecture evolution of AI systems in the era of foundation models, from "foundation-model-as-a-connector" to "foundation-model-as-a-monolithic architecture". The paper then identifies the key design decision points and proposes a pattern-oriented reference architecture to provide reusable responsible-AI-by-design architectural solutions to address the new architecture evolution and responsible AI challenges. The patterns can be embedded as product features of foundation model-based AI systems and can enable organisations to capitalise on the potential of foundation models while minimising associated risks.

Key terms - Responsible AI, ethical AI, architecture, design pattern, design decision, foundation model, large language model (LLM), ChatGPT.

1 Introduction

The release of ChatGPT, Bard, and other large language model (LLM)-based chatbots has drawn huge attention on foundations models worldwide. Foundation models, such as LLMs, are massive AI models that are pre-trained on vast amounts of broad data and can be adapted to perform a wide variety of tasks [1]. With numerous projects already underway to explore their potential, it is widely predicted that foundation model will serve as the fundamental building blocks for most future AI systems.

However, the black box nature and rapid advancements in super-intelligence of foundation models, like ChatGPT, have raised significant concerns around responsible AI. While many ethical principles and frameworks released recently to address these concerns, they lack practical guidance for developers to operationalise them effectively. Furthermore, significant efforts to mitigate AI risks often focus on algorithm-level solutions which only address a subset of ethical principles, such as privacy and fairness. Responsible AI issues can occur beyond algorithms and models and impact various AI and non-AI components of AI systems [2]. Additionally, the foundation model's growing capabilities can eventually absorb the other components of AI systems, introducing the moving boundary and interface evolution challenges in architecture design.

[*]Contact email: qinghua.lu@data61.csiro.au

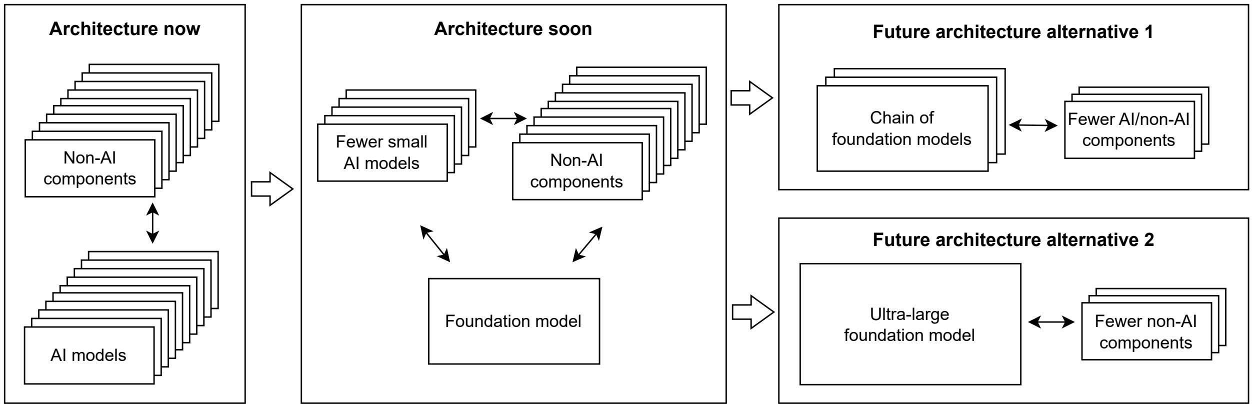

Figure 1: Architecture evolution: from "foundation model as a coordination connector" to "foundation model as a monolithic architecture".

There is an urgent need for concrete system-level guidance to design foundation model-based AI systems. In this paper, we first discuss the potential architecture evolution of AI systems in the era of foundation models and highlight the key quality attributes necessary for the design of responsible foundation model-based AI systems. We then identify the major decision making points when designing foundation model-based AI systems. Finally, we propose a pattern-oriented reference architecture, which provides a responsible-AI-by-design template architectural solution for designing foundation model-based AI systems and considers the evolution of architecture to ensure adaptability over time.

2 Architecture evolution of AI systems

Foundation models are designed to provide a wide range of comprehensive capabilities that can be applied to various tasks, rather than being limited to specific functionalities [1]. One key challenge that the architecture design of AI systems faces with foundation models is that foundation models can eventually absorb the external components such as system functionalities (e.g. prompt components) and software engineering tools. While these components may exist for a while, they can become short-lived and eventually get integrated into the foundation model, resulting a single, monolithic blob at the center of the architecture. As illustrated in Fig.1, the architecture evolution of AI systems can be divided into three stages:

- **Architecture now: many AI models + many non-AI components**: The current architecture of AI systems usually comprises AI models (i.e., AI components) and non-AI components. These AI models and non-AI components co-exist within the architecture of the AI systems and interact with each other to enable the systems to function properly. The AI models are responsible for processing data and making inference, while the non-AI components are responsible for tasks such as user interface, data storage, interaction with other systems.
- **Architecture soon - foundation model as a coordination connector: 1 foundation model + fewer small AI models + many non-AI components**: In this architecture, the foundation model acts as a connector between external components, i.e., small AI models or non-AI components. The role that is fulfilled by the foundation model as a connector is coordination [3]. The foundation model proactively calls the APIs provided by third-party applications to perform tasks. For example, GPT can be responsible for deciding if to call certain tools for certain problems. In such architecture, prompt engineering is important at the early version to guide the foundation models to generate high quality responses. However, it will be eventually absorbed into the foundation model and disappear in the end.
- **Future architecture**:
  - **Alternative 1: chain of foundation models + fewer AI and non-AI components**: There is a chance that most of the AI and non-AI components in Architecture Soon could be absorbed into the foundation models. Thus, one future architecture alternative is a modularised architecture, such as Socratic Models [4]. This architecture relies on a few foundation models that are chained together and a limited number of AI and non-AI components to perform tasks (e.g., through language-based interactions) without requiring additional training or fine-tuning. The inference for a task-specific output is jointly performed by multimodal interactions between the independent foundation models, such as LLMs, visual LLMs and audio LLMs. Those foundation models can be connected via APIs with external AI or non-AI components

that offer additional capabilities or access to databases, such as robotic systems or web search engines. By using task formulation and multimodal interaction between independent models, the architecture can effectively leverage the capabilities of different foundation models and external AI and non-AI components.

- **Alternative 2 - foundation model as a standalone component: 1 ultra-large foundation model**: Another potential type of future architecture is a monolithic architecture, which only contains a single big foundation model capable of performing a variety of tasks by incorporating different types of sensor data for cross-training. An example of this type of architecture is PaLM-E [5], which is used for performing language, visual-language, and reasoning tasks. In this type of architecture, no external components are required, including prompt components.

In the context of this potential architecture evolution, adaptability and modifiability are the two key concerned software quality attributes. *Adaptability* refers to a software system's ability to adapt to run-time changes in its environment without requiring external intervention [6], such as changes in the data being processed. *Modifiability* is the ease with which a software system can be changed at static-time [7], such as adding new features, fixing bugs, or changing the underlying infrastructure. Both adaptability and modifiability are important qualities attributes for an evolving architecture, as they can significantly impact the long-term maintainability of a system. The patterns and tactics of conventional software systems [7, 8] could be applied to manage the issues of moving boundary and interface evolution in the foundation model-based AI systems.

3 Architectural Design decisions

There are some major architectural design decisions that developers need to consider when building foundation model-based systems.

3.1 Design decision 1: Different design options for using foundation models

When designing the architecture, one of the most important decisions is choosing which type of foundation model to use. There could be seven design options:

- **Foundation model type 1**: pre-trained by an external organisation using large unlabeled general data (e.g., general text corpus)
  - **Option 1**: use the foundation model type 1 via in-context learning
  - **Option 2**: fine-tune the foundation model type 1 using labeled target data
- **Foundation model type 2**: first pre-trained using large unlabeled general data, then pre-trained using large unlabeled domain specific data (e.g., public real-estate data). The training can be conducted by the same external organisation or two different external organisations.
  - **Option 3**: use the foundation model type 2 via in-context learning
  - **Option 4**: fine-tune the foundation model type 2 using labeled target data
- **Foundation model type 3**: pre-trained by an external organisation using both large unlabeled general data and large unlabeled domain-specific data
  - **Option 5**: use the foundation model type 3 via in-context learning
  - **Option 6**: fine-tune the foundation model type 3 using labeled target data
- **Foundation model type 4** & **Option 7**: sovereign foundation model which is trained from scratch within an organisation using large unlabeled/labeled general data and/or large unlabeled/labeled domain-specific data

Option 1, 3, 5 can save costs, as they are pre-trained by external organisations on vast amounts of data using numerous computational resources. However, these options may pose responsible AI issues as the pre-training is conducted towards generating generic outputs rather than being designed to a specific purpose. This could result in reliability and ethical issues with the model's outputs. Furthermore, there could be data privacy issues. For example, the data used to train these models may be biased or include personal information. The big tech companies may use the data collected from these models for their own purposes, such as improving their products or developing new features, which may raise privacy concerns.

To improve accuracy and data privacy of the foundation model, Option 2, 4, 6 can be selected, i.e., to locally fine-tune the model with labeled target data (e.g., labeled domain-specific data). However, responsible AI issues may still exist as organisations cannot control the training data and foundation model training process of the external organisations.

To have complete control over data and model training and ensure responsible AI, Option 7 - sovereign foundation model is the best option but it requires high investment in cost and resources, including data, computational, and human resources.

3.2 Design decision 2: Chain of foundation models vs. ultra-large foundation model

When considering foundation models developed by external organisations, one important decision is whether to use a chain of models (such as Socratic Models [4]) or an ultra-large foundation model (such as PaLM-E [5]). The chain of foundation models generates joint predictions and offers an modularised architecture that allows for easy switching to other foundation models with specific capabilities, e.g., switching to a more powerful visual language model to improve performance. This option may improve maintainability, but it may come with an additional cost to understand the capabilities and limitations of different foundations models. On the other hand, using ultra-large foundation model may achieve better performance via cross-training on numerous multi-modal data. However, this option may come at the cost of reduced maintainability. There may be a risk for vendor lock-in, as there may be few providers in the market with similar capabilities. It is challenging to determine which option is better, and experimentation is necessary to evaluate each option's effectiveness for a specific context.

3.3 Design decision 3: Responsibilities of external components

Responsibility is a concept in a software context that comes from object-oriented design[9]. A responsibility can be an action, a piece of knowledge to be maintained, or a decision to be carried out by a software component.

Foundation models can gradually absorb external components by taking on their responsibilities over time. This can create a moving boundary issue where the responsibility of a software component shifts from an external component to the foundation model. To address this issue, one key design decision is to determine the responsibilities of software components. The responsibility can be split into a bunch of smaller responsibilities that are placed in distinct components. Changes can be isolated to specific components, making it easier to manage the external component that could be absorbed by the foundation model over time. As foundation models are built around capabilities [1], it may be worth breaking down a large component along capability lines. This allows the developers to choose which foundation model's capabilities to use, e.g., use a good enough one or an emerging new one.

Breaking down responsibilities into smaller components can improve adaptability and modifiability, ensuring long-term maintainability. However, it can also introduce additional communication overhead between smaller components, as each component may need to interact with other components to accomplish tasks. Additionally, it can make the system more complex and difficult to understand how the components work together, potentially reducing maintainability.

3.4 Design decision 4: Automatic response vs. verifier-in-the-loop

When designing foundation model-based AI systems, an important consideration is how to ensure the systems' responses are accurate and responsible. One option is to rely solely on the foundation model to generate responses to user queries. While this option can be efficient and cost-effective, it may result in inaccurate or irresponsible responses that can affect user trust or cause harm.

To address this issue, a verifier-in-the-loop option can be used, which involves verifying the responses generated by the foundation model before they are sent to the user. This can be done through various means, including human verification, or knowledge data-based verification, or verification by another AI system. While this can help ensure the responses are accurate and responsible, it can be more time-consuming and resource intensive.

The choices between automatic response and verifier-in-the-loop depends on the system's priorities and the consequences of inaccurate or irresponsible responses. For systems where accuracy and trustworthiness are critical, a verifier-in-the-loop approach may be the better option, even if it comes at a higher cost. However, for systems where efficiency and cost-effectiveness are more important, an automatic response approach may be more suitable, with periodic checks.

3.5 Design decision 5: Selection of prompt patterns

Prompt engineering is an important process that involves creating prompts, questions, or instructions to guide the output of a foundation model, like ChatGPT, that is tailored to their specific needs. To ensure that the outputs generated by foundation models are of high-quality and responsible, it is important to use appropriate prompt patterns. There are various prompt patterns available, such as zero/one/few-shot prompt, retrieval/crawling-augmented prompt, negative prompt, multiple choice prompt, chain of thought, etc. The selection of prompt patterns should be based on a variety of factors, such as the system's goal, target users, context, specific tasks being performed. Each prompt pattern may

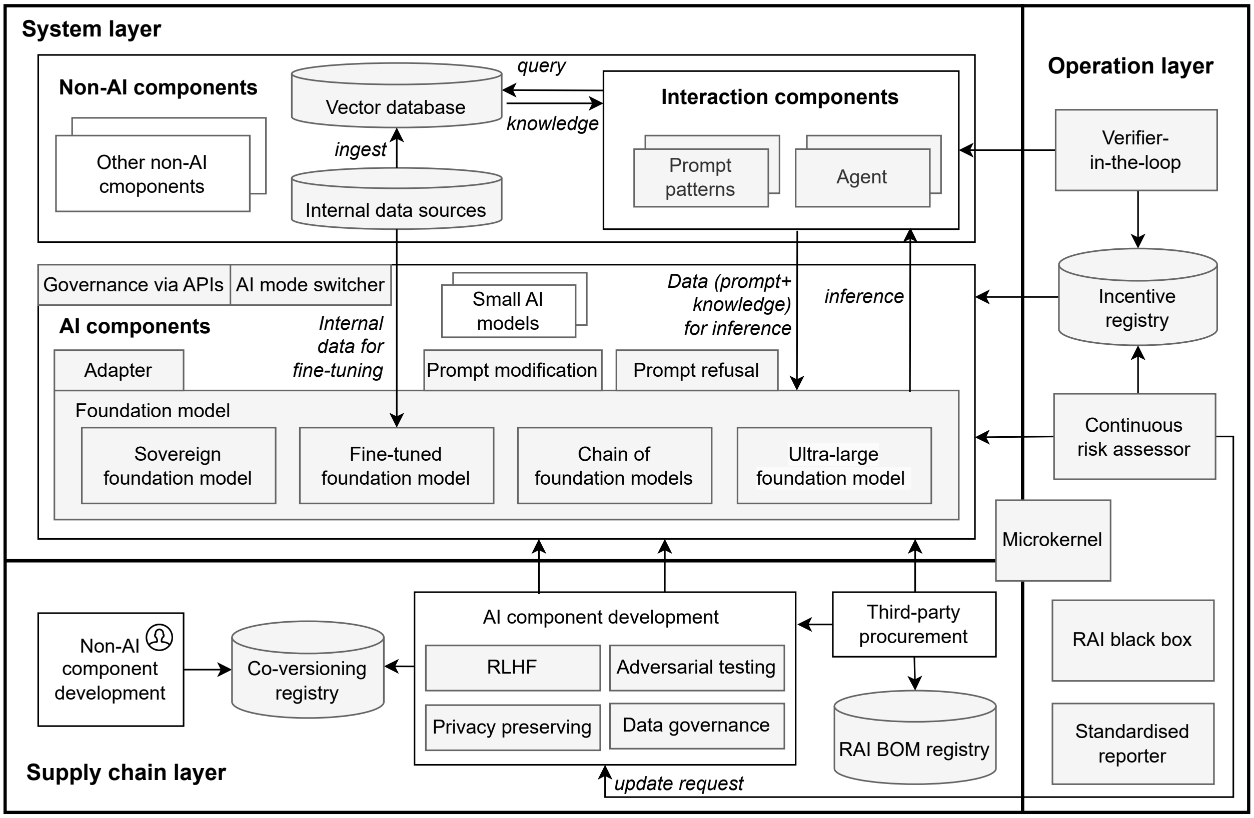

Figure 2: A pattern-oriented reference architecture for designing responsible foundation model-based AI systems.

require different costs and have varying levels of complexity. As foundation models continue to evolve, prompt patterns may become short-lived and obsolete.

3.6 Design decision 6: Single bot vs. bot team

The tasks that need to be performed by a foundation model can vary in complexity and scope. For simple tasks, such as answering frequently asked questions, a single bot instance may be sufficient. However, for more complex tasks, it may be necessary to use a bot team, i.e., multiple bot instances, to ensure the performance. For example, one bot instance can be responsible for question-answering interaction, while another bot instance can be responsible for summarising the documents shared by users. However, using multiple bot instances can also introduce communication costs between bots and increase the level of design complexity.

3.7 Design decision 7: Think aloud vs. think silently

There are two options to consider when it comes to the explaining the decision-making process: think aloud and think silently. The think aloud design pattern can be used disclose the decision-making process, such as the intermediate steps such as prompt pattern implementation and verification/validation. This design can help build human trust in the system, but it may sacrifice data privacy. For example, some system providers may view the prompt design and verification/validation as business sensitive data and intelligence property. In such case, they may need to carefully consider which parts of the intermediate process they are willing to share with the users.

4 Reference architecture

Fig. 2 illustrates a pattern-oriented reference architecture for designing responsible and adaptable foundation model-based AI systems. The architecture comprises three layers: the system layer, which includes the components of the deployed AI system, the operation layer, which provides responsible AI tooling functions to the AI system, and the supply chain layer, which generates the software components that compose the AI system. The grey-coloured boxes and cylinders are the components where the design patterns are applied. The patterns are collected through a multivocal literature review [10, 11] and project experience [2].

4.1 Supply chain layer

The supply chain layer includes all components involved in developing and procuring both AI components (including foundation models) and non-AI components. All components procured from third parties can be associated with a bill of materials (BOM) that records their supply chain details, which can include responsible AI (RAI) metrics or verifiable RAI credentials. This procurement information can be maintained in an **RAI BOM registry**.

Federated learning and other **privacy-preserving techniques** (such as differential privacy and homomorphic encryption) can be used to safeguard the privacy of sensitive data. For example, when fine-tuning the foundation model, the data from an organisation's customers needs to protected through privacy-preserving techniques. To ensure auditability, the **co-versioning registry** pattern can be applied to co-version the AI artifacts, such as foundation models, fine-tuned models, or distilled small models.

4.2 System layer

The system layer consists of AI components and non-AI components. The **foundation model** is a crucial component of the AI system and can be viewed as a design pattern. There are several design patterns for the foundation model (Design decision 1 & 2), including the **fine-tuned foundation model**, **sovereign foundation model**, **chain of foundation models**, **ultra-large foundation model**. Using a **foundation model** from a big tech company can save costs, but may result in reliability, ethical, and data privacy concerns. To address these issues, **fine-tuned foundation model** pattern can be used to retrain the foundation model locally with domain-specific knowledge, but responsible AI issues may still exist . The **sovereign foundation model** pattern can be used to have full ownership and ensure responsible AI, but it requires high investment. The **chain of foundation models** pattern allows for easy switching to other foundation models with specific capabilities, improving maintainability but at an additional cost to understand the capabilities and limitations of different foundations models. The ultra-large foundation model pattern can achieve better performance via cross-training on numerous multi-modal data, but may reduce maintainability and create vendor lock-in.

Due to the rapidly growing capabilities of foundation models, there are issues with the moving boundary and interface evolution in architecture design. Most of the components in the system layer, operation layer, and supply chain layer will eventually be absorbed by foundation models. In some cases, absorption is not done component by component, but rather by splitting a component in the middle. To address these issues, two patterns can be applied to ensure adaptability and modifiability: **microkernel** pattern [7] and **adapter** pattern [12] (also known as **wrapper**). **microkernel** pattern (Design decision 3) can help place smaller responsibilities in distinct component so that changes can be isolated to specific components, which could be absorbed by the foundation model overtime. When a component is absorbed by a foundation model, the component's original connector used to communicate with other components may need to be converted into a certain format of interface (such as a text interface for LLMs) through the use of an **adapter** pattern. The **bot team** pattern (Design decision 6) can be used to perform complex tasks, splitting the task into a few small tasks and assigning them to different bot instances.

To prevent harmful dual-use of AI systems, developers should impose restrictions on their usage and prevent users from getting round of restrictions through unauthorised reverse engineering or modification of the system design. One way to do this is by implementing **governance via APIs** pattern, which involves providing AI services on cloud platforms and managing interactions through API controls, rather than allowing AI systems to run locally with unrestricted access. **AI mode switcher** pattern can be used to decide when to switch from automatic response to **verifier-in-the-loop**.

The **domain-specific knowledge base** pattern can used to 1) fine-tune the foundation model; 2) provide additional knowledge data for inference; 3) verify/validate/explain the responses. The knowledge base could internal business data or external domain data. The **prompt patterns** (Design decision 5) are commonly used in the interaction components to improve the input quality of the foundation model and guide its responses. **Think aloud** pattern can be employed to make the intermediate process transparent and improve human trust in AI systems.

4.3 Operation layer

Verifier-in-the-loop pattern is particularly useful when accuracy and trustworthiness are critical. In this pattern, a verifier is responsible for verifying or modifying the responses returned by the foundation model, or providing feedback

[2]`https://research.csiro.au/ai4m/operationalising-responsible-ai/`

to agree or disagree with the responses. The verifier can be a human (e.g., a domain expert or user), or a knowledge data-based verification mechanism, or another AI system.

Incentive mechanisms are effective treatments in motivating AI systems and encouraging the stakeholders to execute tasks in a responsible manner. The **incentive registry** pattern can be used to record the rewards that correspond to the AI system's ethical behavior and outcome of decisions. There are various ways to formulate the incentive mechanism, such as using reinforcement learning or building it on a publicly accessible data infrastructure like blockchain.

The **continuous risk assessor** pattern can be used to monitor and validate the outcomes of AI systems against RAI metrics specified by different responsible AI frameworks and standards. The outcomes of AI systems include the result of the AI component, and the impact on users and the broader ecosystem. The ethical impact of the system behavior, as the output of the **continuous risk assessor**, is the input to the **incentive registry**, which records the rewards that correspond to the AI system's ethical behavior and outcome of decisions.

The **RAI black box** pattern is crucial to ensure accountability and auditability of foundation model-based AI systems. By recording critical data in an immutable data ledge (such as blockchain), the RAI black box allows for accountability analysis after near misses and incidents. This includes data such as the input and output of foundation models and small AI models, the versions of foundation models and small (distilled) AI models, etc.

The **standardised reporter** pattern can be used to inform stakeholders (such as regulators and users) about the development process and product design of AI systems, such as RAI BOM information about the foundation models and data/model/system card.

5 Conclusion

This paper presents pattern-oriented responsible-AI-by-design reference architecture to address the challenges of responsible AI and architecture evolution in foundation model-based AI systems. We first discuss the architecture evolution and identify two important software qualities for building foundation model-0based AI systems: adaptability and modifiability. Then, we summarise seven key design decisions in architecture design and discuss the trade-offs between responsible AI related software qualities. Finally, we present a pattern-oriented reference architecture to provide a concrete guidance for developers to design responsible and adaptable foundation model-based AI systems. In the future, we will build a pattern catalogue for building foundation model-based AI systems.

References

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] Q. Lu, L. Zhu, X. Xu, J. Whittle, and Z. Xing, "Towards a roadmap on software engineering for responsible ai," in *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*, 2022, pp. 101–112.

[3] N. Medvidovic and R. N. Taylor, "Software architecture: foundations, theory, and practice," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, 2010, pp. 471–472.

[4] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

[5] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[6] N. Subramanian and L. Chung, "Metrics for software adaptability," *Proc. Software Quality Management (SQM 2001)*, vol. 158, 2001.

[7] F. Bachmann, L. Bass, and R. Nord, "Modifiability tactics," CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, Tech. Rep., 2007.

[8] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[9] R. Wirfs-Brock and A. McKean, *Object design: roles, responsibilities, and collaborations*. Addison-Wesley Professional, 2003.

[10] Q. Lu, L. Zhu, X. Xu, J. Whittle, D. Zowghi, and A. Jacquet, "Responsible ai pattern catalogue: a multivocal literature review," *arXiv preprint arXiv:2209.04963*, 2022.

[11] Q. Lu, L. Zhu, X. Xu, and J. Whittle, "Responsible-ai-by-design: A pattern collection for designing responsible ai systems," *IEEE Software*, 2023.

[12] E. Gamma, R. Johnson, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.