

Large Language Models for Assessing Microservices Architectures: Current and Future Research Directions

MATEUS DUTRA

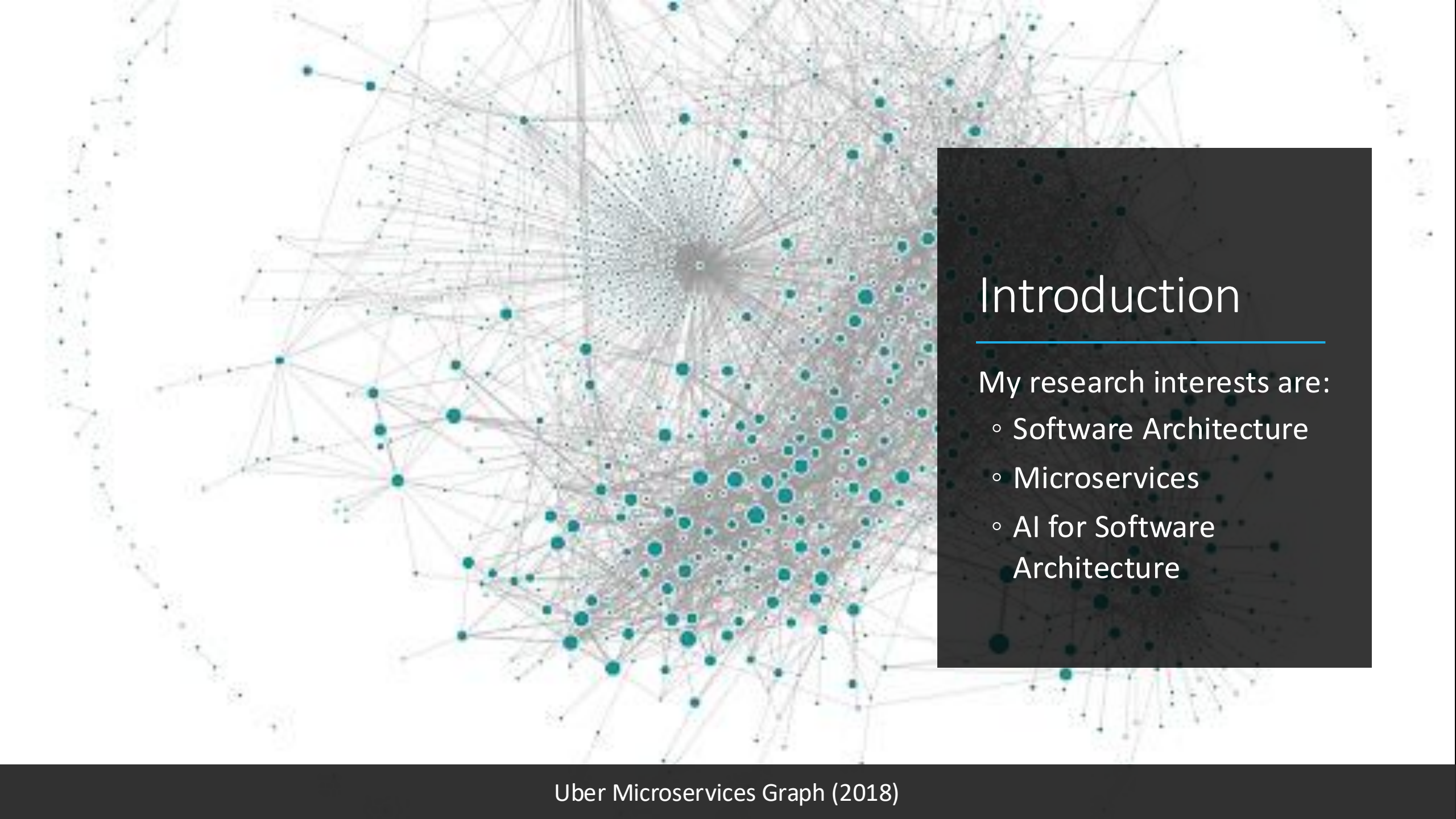
UF *m* G

lab-soft
software engineering laboratory

Introduction

“From Detection to Refactoring of Microservice Bad Smells: A Systematic Literature Review”

- Authors: Mateus Dutra, Denis Rodrigo, Johnatan Oliveira and Eduardo Figueiredo
- Accepted in *Journal of Software Engineering Research and Development* (JSERD)
- Focus: Review of detection tools of microservice bad smells and refactoring strategies

A complex network graph visualization representing the Uber microservices architecture in 2018. The graph consists of numerous nodes, represented by small circles, connected by a dense web of thin lines (edges). The nodes are colored in various shades, including teal, blue, and pink, and are distributed across the frame, with a higher concentration in the center and right side. The overall structure is highly interconnected, showing a complex, multi-layered network.

Introduction

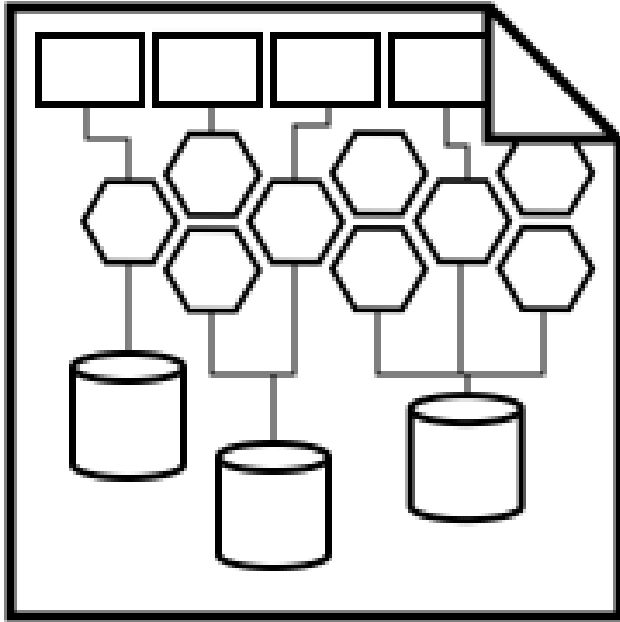
My research interests are:

- Software Architecture
- Microservices
- AI for Software Architecture

Background

“Architecture is about the important stuff. Whatever that is.”, Martin Fowler.

Perry and Wolf [2] define as: *“the system's key elements, and their relationships at each other and to their environment. Furthermore, the architecture reflects the rationale behind the system's structure, functionality, interactions, and resulting properties”*.



Background

As projects grow, the *descriptive architecture* often drifts away from the *prescriptive architecture*, leading to *architecture erosion* [4].

Understanding both the “*as-is*” and the “*to-be*” architectures is essential for making better decisions.

Related Work

Paper	Conference/Journal	Year	Reference
Can LLMs Generate Architectural Design Decisions? - An Exploratory Empirical Study	ICSA	2025	[6]
Do Large Language Models Contain Software Architectural Knowledge? An Exploratory Case Study with GPT	ICSA	2025	[5]
Generative AI for Software Architecture. Applications, Challenges, and Future Directions	JSS	2025	[3]
LLM-Based Quality Assessment of Software Architecture Diagrams: A Preliminary Study with Four Open-Source Projects	EASE	2025	[1]
LLMs for Generation of Architectural Components: An Exploratory Empirical Study in the Serverless World	ICSA	2025	[7]

LLM-Based Quality Assessment of Software Architecture Diagrams: A Preliminary Study with Four Open-Source Projects

Glauber Queiroz de Oliveira and Nabor C. Mendonça^(✉) 

Post Graduate Program in Applied Informatics, University of Fortaleza, Fortaleza,
CE, Brazil

`nabor@unifor.br`

Abstract. This work explores the feasibility and challenges of using a Large Language Model (LLM) to automatically assess the quality of software architecture diagrams. Our approach is based on a structured

LLM-Based Quality Assessment on Software Architecture Diagrams

- Four open-source projects
- Use Chat-GPT as a judge of software architectures diagrams
- Evaluation criteria:
 - *Clarity*
 - *Consistency*
 - *Completeness*
 - *Accuracy*
 - *Level of Detail*



Our Goal

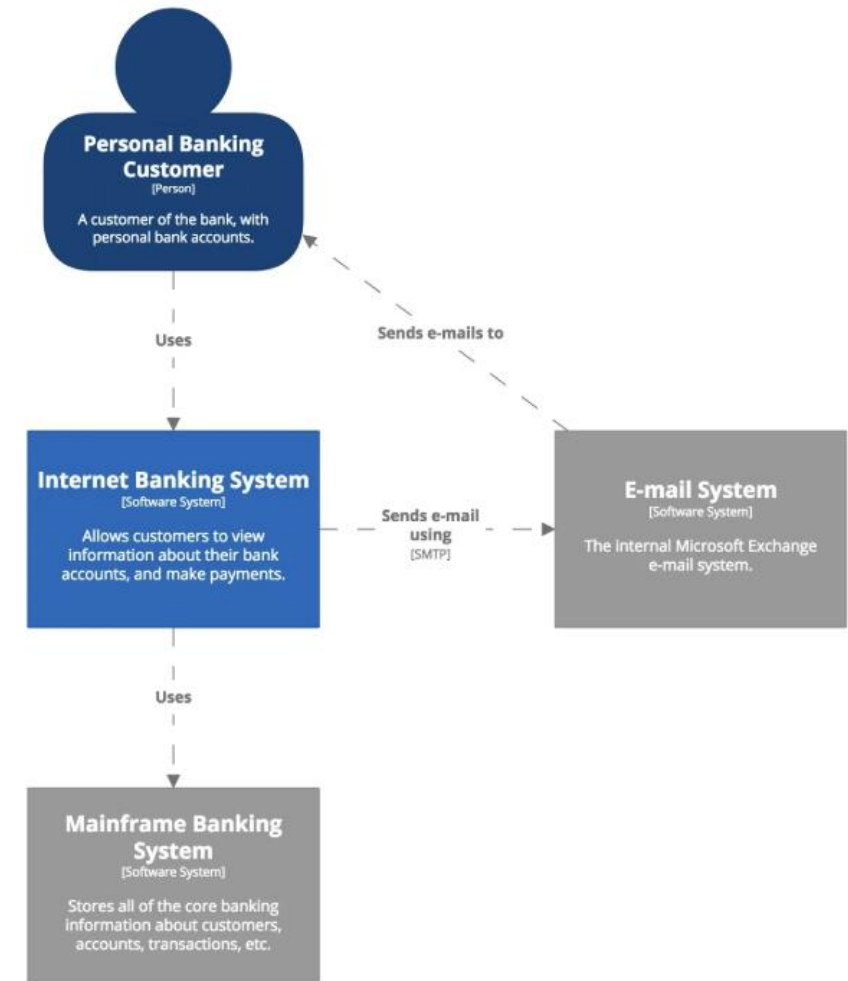
Leverage Large Language Models (LLMs) for Software Architecture support

Possible research questions:

- Can LLMs generate software architecture documentation?
- Can LLMs take design decisions?
- Does the common LLMs have Architectural Knowledge?
- How LMMs evaluates existing architectures?

Methodology

- C4-Model diagrams
- Diagram-as-a-code
- High-Level architecture diagrams
- Evaluate quality attributes
- Compare different LLMs
- Human verification



Architecture Assessment Prompt

Objective: Evaluate the quality of the attached architecture diagram, considering both the diagram and its description below.

[Insert diagram here]

Description: [Insert textual description here]

Evaluation Criteria:

1. *Clarity:* The diagram should be understandable to both technical and non-technical stakeholders. Assess whether the symbols, labels, and information flow are clear. Provide suggestions for improving clarity, if necessary (e.g., using more accessible language, improving labeling, simplifying the explanation of key components, providing a clearer description).
2. *Consistency:* Check whether symbols, styles, and terms are used uniformly throughout the diagram. If there is inconsistency (e.g., symbols representing the same type of component are different), highlight this and recommend a standard set of symbols or styles. Also, check for consistency between the architecture diagram and its description (e.g., are both communicating the same content? Are there any discrepancies or omissions in either the diagram or the description? Provide suggestions for improvement if needed).
3. *Completeness:* The diagram should present all relevant components of the architecture, including

any interactions between them. Note any missing components or interactions and suggest what should be added to provide a comprehensive view.

4. *Accuracy:* Ensure that the diagram accurately reflects the described architecture. Highlight any discrepancies between the diagram and the text (e.g., if a component described in the text is missing in the diagram or vice versa). Provide recommendations to resolve any inaccuracies.

5. *Level of Detail:* Evaluate whether the level of detail is appropriate for the target audience. Technical diagrams may require more details for developers, while high-level diagrams for stakeholders should simplify complex concepts. If the diagram is too detailed or too vague for the intended audience, suggest changes to meet the audience's needs.

Evaluation Instructions:

For each criterion, rate the diagram as follows:

Meets Expectations: No significant changes needed.

Partially Meets Expectations: Minor improvements needed (explain what they are).

Does Not Meet Expectations: Significant issues need to be addressed (explain what they are and provide specific suggestions for improvement).

Use this structure to ensure that the evaluation is thorough and specific. Justify your rating for each criterion and provide suggestions for improvement where necessary.

Methodology (Setup)

- **Dataset:** Four diagrams found online in github
- **LLMs:**
 - *Chat-GPT, Deepseek and Gemini*
- **Prompt:** inspired by Oliveira and Mendonca [1]

Preliminary Findings

- ChatGPT more concise responses
- Deepseek and Gemini generate more tokens
- Deepseek was the most divergent from the others
- Across all criteria, at least two models produced the same outcome
- **Current status:**
 - Validation of models responses (work in progress)

Challenges

- Get a relevant dataset
- Create the ground “truth”
- How to evaluate LLMs responses?



Thank you!

Questions or
Suggestions?

References

- [1] G. Q. de Oliveira and N. C. Mendonça, “LLM-Based Quality Assessment of Software Architecture Diagrams: A Preliminary Study with Four Open-Source Projects,” in *Proc. 19th European Conf. on Software Architecture (ECSA)*, 2025.
- [2] D. E. Perry and A. L. Wolf, “Foundations for the Study of Software Architecture”, in *ACM SIGSOFT Software Engineering Notes*, 17(4), pp. 40–52, Oct. 1992.
- [3] M. Esposito, X. Li, S. Moreschini, N. Ahmad, T. Cerny, K. Vaidhyanathan, V. Lenarduzzi, and D. Taibi, “Generative AI for Software Architecture: Applications, Challenges, and Future Directions” *arXiv preprint arXiv:2503.13310*, 2025.
- [4] N. Medvidovic and R. N. Taylor, “Software Architecture: Foundations, Theory, and Practice,” in *Proc. 32nd ACM/IEEE Int. Conf. on Software Engineering (ICSE)*, Vol. 2, pp. 471–472, 2010.

References

- [5] M. Soliman and J. Keim, “Do Large Language Models Contain Software Architectural Knowledge? An Exploratory Case Study with GPT,” in *Proc. 22nd IEEE Int. Conf. on Software Architecture (ICSA)*, 2025.
- [6] R. Dhar, K. Vaidhyanathan, and V. Varma, “Can LLMs Generate Architectural Design Decisions? An Exploratory Empirical Study,” in *Proc. 21st IEEE Int. Conf. on Software Architecture (ICSA)*, 2024.
- [7] S. Arun, M. Tedla, and K. Vaidhyanathan, “LLMs for Generation of Architectural Components: An Exploratory Empirical Study in the Serverless World,” in *Proc. 22nd IEEE Int. Conf. on Software Architecture (ICSA)*, 2025.