CPT402 Software Architectures
Coursework 1 – Research Report

Start date:	*6 March 2025*
Deadline:	*23:59(Beijing time, China), 10 April 2025*
Learning outcomes assessed:	*ACDEF*
Percentage in final mark:	*50%*
Late submission policy:	*5% of the total marks available for the assessment shall be deducted from the assessment mark for each working day after the submission date, up to a maximum of five working days.*
Risk:	• *Please read the coursework instructions and requirements carefully. Not following these instructions and requirements may result in loss of marks.* • *Plagiarism results in award of ZERO mark; all submissions will be checked by TurnItIn.* • *The formal procedure for submitting coursework at XJTLU is strictly followed. Submission link on Learning Mall will be provided in due course. The submission timestamp on Learning Mall will be used to check late submission.*

1. Description

Large language models (LLMs) have achieved remarkable success and been revolutionising the field of artificial intelligence (AI). These models can compress world knowledge into extraordinarily large neural networks, and generate reasonable solutions on complex reasoning and problem-solving tasks. With a simple interface, they can engage in natural language conversations, answering questions, document summarisation, translation, and generation of images, poems, and even music. They represent the very first class of tools in the research and development of AI that really work in the history.

Many of these LLMs also exhibit strong capabilities in coding and programming tasks. Such models are usually trained with vast amount of existing code related data and can greatly improve efficiency and productivity for programmers. These AI coding assistants can be used in a variety of coding tasks, such as generation, completion, analysis, debugging, refactoring, and testing. Some representative examples include GitHub Copilot [1], [2], CodeQwen [3], Qwen2.5-Coder [4], Code Llama [5], Claude Opus [6], Gemini [7], [8], and GPT 4 [9].

In developing complex, distributed systems beyond simple algorithmic coding, such as enterprise information systems using multi-tiered architectures or microservices, while LLMs can certainly provide great support to developers in every stages of the software development lifecycle [10], e.g. from requirements engineering [11], conceptual modelling [12], system design [13], code generation [14], to testing/verification [15], challenges remain. This can be attributed to the sophisticated nature of distributed system development, for example, complex and iterative customer requirements, model-view-controller (MVC) pattern, complex system configuration and seamless integration of individual functional/non-functional components. How to effectively and efficiently use LLMs in large-scale, distributed system development still needs further study and investigation [16].

2. Requirements

You are asked to study some representative LLMs with strong coding capabilities (such as the ones listed above) and produce a research report on their uses in distributed system development. The report should contain two major sections: (1) a short survey on recent LLMs with strong coding and programming capabilities, and (2) analysis of such LLMs in developing complex distributed systems (e.g. using multi-tiered architectures or microservices), including benefits, challenges and outlook. You may focus on one or more stages of the software development lifecycle (e.g. requirements engineering, conceptual modelling, design, code generation, testing or verification).

You should search for more publications and expand your reading beyond the ones provided in this document, which ONLY serve as a starting point.

Detailed requirements on the research report are specified as follows.

- The report itself should **NOT** exceed **FIVE** pages including everything, e.g. text, tables, diagrams and references (suggested number of references is around 20), with the provided IEEE conference template (***DO NOT alter the original template***).
- References should **NOT** contain survey papers.
- Abstract and Conclusion are **NOT** needed.
- The report should have your affiliation information as in the template.
- The structure below is suggested.
    - Section ONE: Introduction
        - Provide a short background introduction.
        - Define the scope of the study.
    - Section TWO: Survey of LLMs with strong coding and programming support
        - Provide short technical details, strength and limitations, wherever appropriate).
        - Evaluation and comparison based on common benchmarks.
    - Section THREE: LLMs in developing distributed systems – challenges and outlook

- May focus on any of the stages (one or more): requirements engineering, conceptual modelling, system design, code generation, and testing/verification.
- Provide a detailed analysis on the benefits, challenges and prospects of using LLMs as AI assistants in developing complex distributed systems.
- Discuss any legal, social, ethical, environmental and professional issues and implications with the use of artificial intelligence techniques and minimise adverse impacts.

3. Learning Outcomes

After conducting the research, you should be able to:

- Understand and design architectures for complex distributed software systems;
- Critically evaluate distributed systems using relevant technologies;
- Critically analyse the practical and theoretical problems which exist in distributed computing systems;
- Participate in the legal, social, ethical and professional framework in the development of distributed systems;
- Analyse the influence of artificial intelligence techniques and evaluate their environmental and societal impact on the solutions to sophisticated distributed systems, and minimise adverse impacts.

4. Deliverables

- A research report according to the requirements set out in Section 2 of this document.
- Convert report into **PDF**, and name it as "***CPT402-CW1-YOUR-ID.pdf***".

References

1. GitHub Copilot, Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. ArXiv, abs/2302.06590. https://arxiv.org/pdf/2302.06590
2. Dakhel, A.M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M.C., & Jiang, Z.M. (2022). GitHub Copilot AI pair programmer: Asset or Liability? J. Syst. Softw., 203, 111734.
3. Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, J., Yang, S., Yao, Y., Yu, B., Bowen, Y., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., & Zhu, T. (2023). Qwen Technical Report. ArXiv, abs/2309.16609.

4. Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Dang, K., Yang, A., Men, R., Huang, F., Quan, S., Ren, X., Ren, X., Zhou, J., & Lin, J. (2024). Qwen2.5-Coder Technical Report. ArXiv, abs/2409.12186.
5. Baptiste Rozie`re, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Je´re´my Rapin, et al. Code Llama: Open foundation models for code. arXiv preprint arXiv:2308.12950, 2023.
6. The Claude 3 Model Family: Opus, Sonnet, Haiku., https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf
7. Rohan et al. Gemini: A Family of Highly Capable Multimodal Models. CoRR abs/2312.11805 (2023)
8. Reid et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. ArXiv, abs/2403.05530. (2024)
9. OpenAI. GPT4 technical report. arXiv preprint arXiv:2303.08774, 2023.
10. Belzner, L., Gabor, T., Wirsing, M. (2024). Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study. In: Steffen, B. (eds) Bridging the Gap Between AI and Reality. AISoLA 2023. Lecture Notes in Computer Science, vol 14380. Springer, Cham.
11. Görgen, Leon, Eric Müller, Marcus Triller, Benjamin Nast, and Kurt Sandkuhl. "Large Language Models in Enterprise Modeling: Case Study and Experiences.", 12th International Conference on model-based software and systems engineering, 2024.
12. Fill, Hans-Georg, Peter Fettke, and Julius Köpke. "Conceptual modeling and large language models: impressions from first experiments with ChatGPT." Enterprise Modelling and Information Systems Architectures (EMISAJ) 18 (2023): 1-15.
13. White, Jules, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C. Schmidt. "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design." arXiv preprint arXiv:2303.07839 (2023).
14. Lahiri, S.K., et al.: Interactive code generation via test-driven user-intent formalization. CoRR, abs/2208.05950
15. Liu, J., Xia, C.S., Wang, Y., Zhang, L.: Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. CoRR, (2023)
16. C. Ebert and P. Louridas, "Generative AI for Software Practitioners," in IEEE Software, vol. 40, no. 4, pp. 30-38, July-Aug. 2023