Will Large Language Models Transform Software Engineering?

An Analysis of the Benefits and Challenges of the Application of Large Language Models to Software Engineering

Jake West-Gomila

June 29, 2025

Abstract

Software is a critical technology in modern society (Meisel, 2013) and the application of large language models (LLMs) to software engineering (SE) is of significant consequence to the technology industry and beyond (Hou et al., 2024; Bommasani et al., 2021). This essay weighs the benefits of the application of LLMs to SE against the risks and finds that, despite facing many and varied challenges, LLMs will transform SE in the coming years.

1 Introduction

Large language models (LLMs) are used by millions of people every day, directly through applications like ChatGPT (Rainie, 2025) and indirectly through services like Google Search, which use Google's Gemini models for AI overviews (Reid, 2024). LLMs have significant economic potential because they can automate many language-based cognitive tasks (Bommasani et al., 2021; Yang et al., 2022), from text summarisation and question answering, to sentiment analysis and coding. General purpose LLMs achieve a high level of performance on unseen code generation and completion tasks (White et al., 2025; Jain et al., 2024), two of the core responsibilities of software engineers (SWE). Given the existing capability of LLMs to perform some software engineering (SE) tasks (Chen et al., 2021), it is pertinent to acknowledge that more powerful, domain specialised or otherwise augmented models may arise that can perform at human level, or beyond, across a broad range of such tasks.

2 Background

The transformer network architecture transformed the field of artificial intelligence by enabling LLMs (Vaswani et al., 2017). The key innovation was a mechanism called self-attention, which unlocked efficient long-distance context modelling (Russell and Norvig, 2022), significantly reducing the cost of model training while increasing accuracy (Vaswani et al., 2017).

Decoder-only transformer models are the predominant architecture in both LLM research and production systems (Hou et al., 2024; Meta AI, 2024). These models generate outputs autoregressively, producing each token directly from the sequence of input and previously generated tokens (Vaswani et al., 2017). LLMs are trained on large-scale text and code data, and they implicitly learn code syntax and problem-solving patterns, enabling them to translate natural language descriptions into executable code (Hou et al., 2024). Their coding ability is fine-tuned by reinforcement learning from human feedback, in which human evaluators rank LLM output and a reward model updates the LLM parameters to preferentially output human-like code (Stiennon et al., 2020).

SE is a discipline concerned with the development and maintenance of software systems, and LLMs are especially effective at tasks like code completion (Hou et al., 2024). However, it is the ability to remodel SE challenges into code or text analysis tasks that means SE is a key beneficiary of the LLM revolution (Hou et al., 2024).

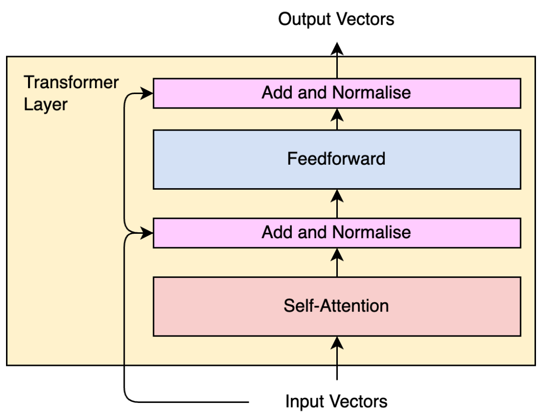

Figure 1: A transformer layer has three distinct components, self-attention, a feedforward network and residual connections (adapted from Russell and Norvig, 2022).

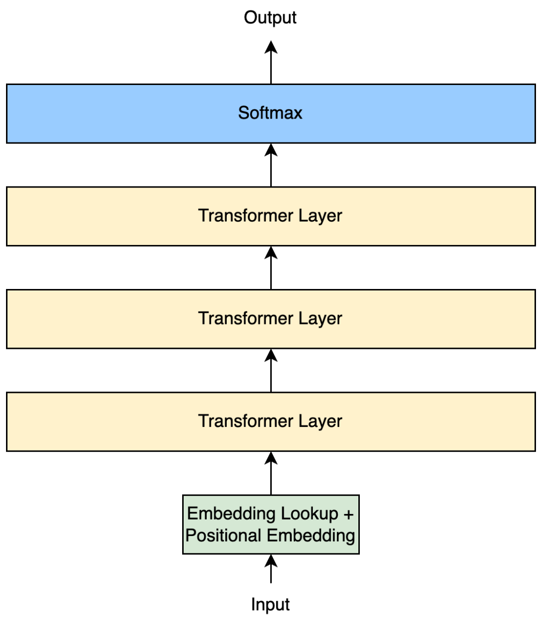

Figure 2: An example transformer network consisting of an embeddings component, three transformer layers and a softmax layer to predict the next token (adapted from Russell and Norvig, 2022). Inputs are converted into embeddings and passed through multiple transformer layers (each containing attention and feedforward sublayers), with each hidden layer building a richer internal representation of the input (Vaswani et al., 2017). State of the art LLMs have dozens of hidden layers and hundreds of billions of parameters (Brown et al., 2020).

3 Applications to Software Engineering

3.1 The Six Domains

The application of LLMs to the six domains of SE tasks have been researched to varying degrees (Figure 3), and Hou et al.'s literature review demonstrates that LLMs have the potential to impact all domains (Figure 5).

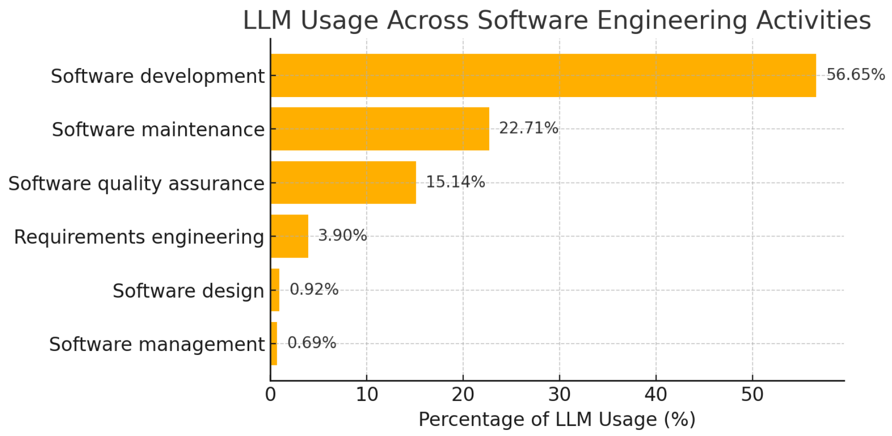

Figure 3: LLM usage across SE activities (adapted from Hou et al., 2024). Software development dominates recent research efforts, with domains like software design receiving relatively little focus.

Software management and design are the least researched domains, but critical tasks like effort estimation and system design have shown promise (Hou et al., 2024). Requirements engineering is more popular and the capability of LLMs to perform tasks like specification generation reinforce their potential to benefit the entire software development lifecycle (Figure 4; Hou et al., 2024).

3.2 Software Development

Software development is by far the most researched domain, and LLMs display advanced code generation capabilities in certain contexts (Chen et al., 2021). Through applications like ChatGPT, LLMs have started to democratise SE by enabling effective method-level code generation and completion (Hou et al., 2024). However, they become less effective as the degrees of separation between the problem specification and the code increase (Chen et al., 2021), with LLM performance on class-level code generation tasks being much worse than method-level (Du et al., 2023). LLMs require human SWEs to define granular tasks and moderate output, which means the more significant milestone of whole program synthesis, in which an LLM creates significant parts of an application from scratch, is not yet possible (Cognition AI, 2024).

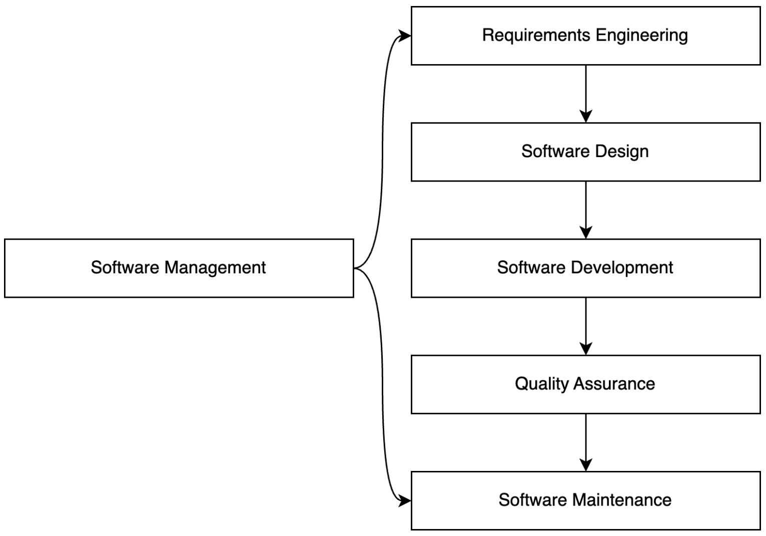

Figure 4: The software development lifecycle (SDLC) constructed from the six SE task domains (Hou et al., 2024) demonstrates the broad potential for LLMs to impact SE. Software management is the outlier, impacting two domains due to its dual tasks of effort estimation (requirements engineering) and tool configuration (software maintenance).

3.3 Software Quality Assurance and Maintenance

Software quality assurance and maintenance go hand in hand with software development by maintaining the reliability, security and performance of software systems through tasks like testing and software version updates. Gomes, da Silva Torres and Côrtes found BERT models to be effective in predicting long lived bugs which can reduce software quality and user sentiment (2023). For when bugs reach production, Jin et al. developed Interfix, a novel encoder-decoder LLM system fine-tuned for program repair (2023). Erroneous software can have significant physical and financial consequences (Zhivich and Cunningham, 2009), therefore, these tools are valuable to society.

3.4 API Synthesis

LLMs can bridge human intent and machine execution by synthesising API calls from natural language descriptions and acting on the results (Hou et al., 2024), enabling complex programs to be constructed with zero code while reducing bugs caused by hallucinations. In the future, software users may only need a single interface, the interface to their favoured LLM, to interact with a multitude of systems. Such a development affects SE in two significant ways. Firstly, sub-professions such as user interface developers may become obsolete; this raises employment and diversity risks. Secondly, SE would shift toward an API-first paradigm, allowing companies to deliver value solely through feature rich, reliable, secure and performant server-side solutions.

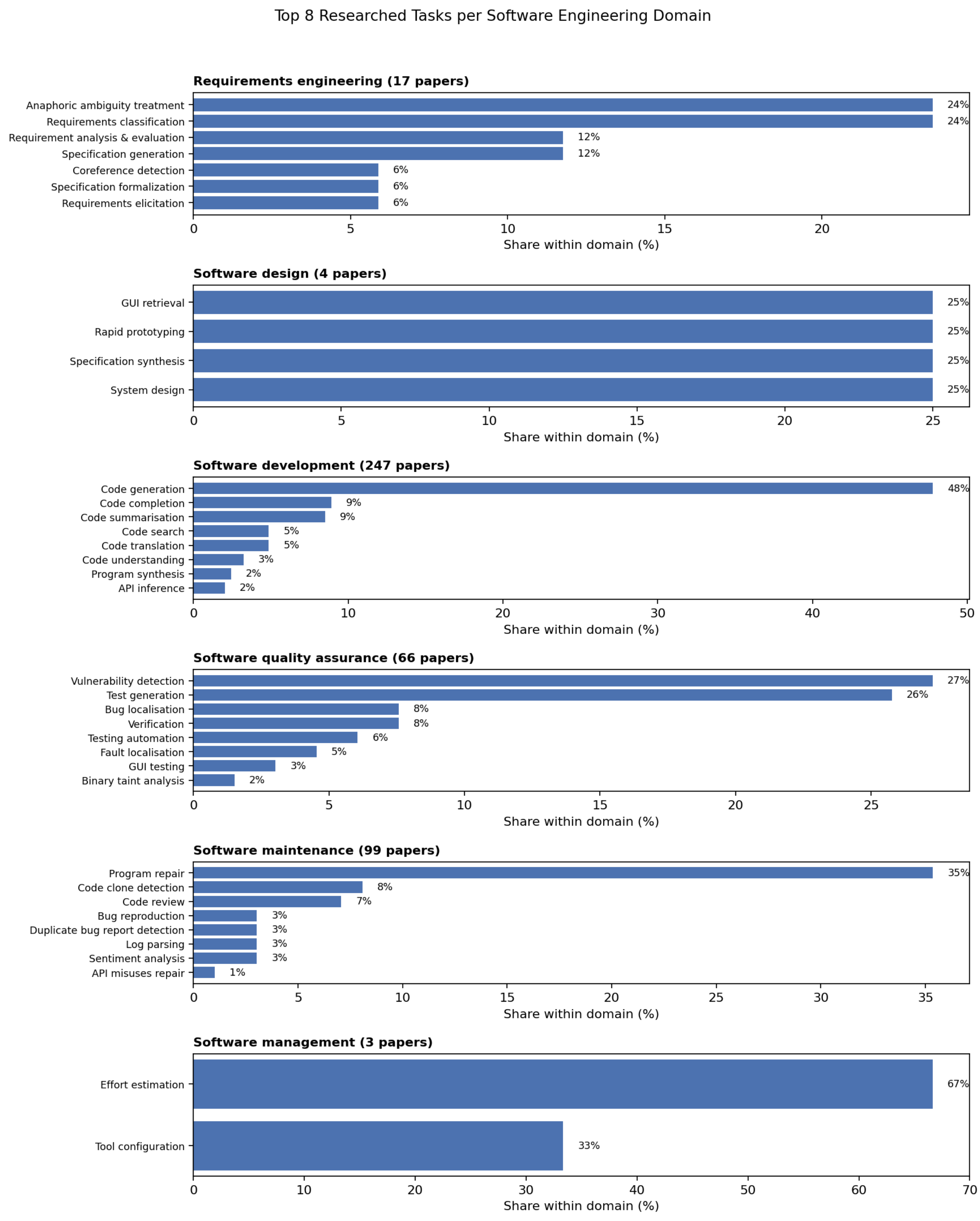

Figure 5: Shows the top 10 tasks within each SE domain and their percentage representation in the literature (adapted from Hou et al., 2024). Note the frequent dominance of one or two tasks within each domain, raising the prospect of advancements in the others.

3.5 Productivity and Suitability

LLMs significantly increase SE productivity (GitHub, 2024), and LLM powered AlphaEvolve can improve its own underlying algorithms (AlphaEvolve Team, 2025); it is plausible that LLMs applied to SE will accelerate the development of LLM systems themselves, creating a positive feedback loop (Anthropic, 2025). Conversely, the rapid advancement of these systems leaves less time to ensure their suitability for mass deployment; critical prompt injection security vulnerabilities have already been discovered in state-of-the-art LLM systems (CVE Program, 2025).

4 Risks to Software Engineering

4.1 Employment Risk

Capitalist market economies account for the majority of global output and employment (Sachs, 1999). Within these economies, competition and corporate governance orient firms toward the maximisation of shareholder value (Friedman, 1970). LLM-driven increases in productivity imply a reduction in the unit cost of SE. When combined with market forces, this means firms will be incentivised to reduce their reliance on human SWEs. Such changes to the industry's workforce constitution could have myriad implications, particularly if cooccurring with broader labour force shifts, such as unionisation of SWEs or state regulation to artificially maintain human employment.

4.2 Algorithmic Bias

Capitalist markets disincentivise actions not directly aligned with maximising shareholder value (Friedman, 1970), therefore, the development of ethical systems is likely to be a secondary concern for companies. LLMs are prone to bias (Buyl et al., 2024) and as their use in SE grows, so does the risk of this bias propagating into software systems. For example, algorithmic bias negatively impacts the ability of autonomous vehicles (AV) to detect people with darker skin tones (Wilson, Hoffman, and Morgenstern, 2019). If LLM code synthesis were used to help build a new AV object detection system, it may propagate the same biases to it. Governments must establish the correct incentive structures for companies because code lays the foundations for world-changing systems; the generation of biased code by LLMs has the potential to cause allocative and representational harm at scale (Chen et al., 2021).

4.3 Diversity Concerns

LLMs encode the biases of their creators, from the training corpora selected to application operational parameters such as guardrails (Buyl et al., 2024). Most state-of-the-art LLMs are built by researchers who are predominantly Western, Educated, Industrialized, Rich, and Democratic (WEIRD; Henrich, Heine, and Norenzayan, 2010), a concentration noted by West, Whittaker & Crawford (2019). Therefore, by replacing SWEs with LLMs, firms risk shrinking both workforce and cognitive diversity; DEI-focused hiring is one option to counter the resulting algorithmic bias.

LLMs tend to reduce content diversity, which is detrimental to personal expression and creativity (Padmakumar and He, 2024). Furthermore, homogenised content may be propagated as future models are trained on it, creating a positive feedback loop (Padmakumar and He, 2024). This challenge could impact SE through, for example, the propagation of popular but flawed patterns such as premature optimisation. Preventing models from training on LLM-generated data would break the feedback loop and prevent this mode of content homogenisation. Given the proliferation of LLMs, a future solution to this problem will likely depend on digital watermarking (Cohen, Hoover, and Schoenbach, 2024).

4.4 Content Ownership Challenges

LLMs are trained on diverse data sources, including copyrighted material, and are capable of recreating copyrighted code from their training data (Cooper et al., 2025). SWEs may inadvertently infringe on the copyright of another entity, resulting in legal risk (Cooper et al., 2025). Depending on the rulings of ongoing copyright lawsuits (Baker & Hostetler LLP, 2025), companies may be incentivised to train LLMs using corpora excluding copyrighted works. However, an arguably more significant barrier to adoption is that using LLMs for SE tasks exposes proprietary code to the organisation running the LLM, which is in turn used as training data for future models (OpenAI, 2025). Running large open-source LLMs on private servers could help companies safeguard their data, but this solution is complex, costly and potentially infeasible for smaller firms.

5 Conclusion

This essay has discussed risks relating to employment, bias, diversity, content ownership and suitability. Modern society is dependent on software systems, and software systems depend on good engineering; the misapplication of LLMs to SE could enable physical and financial harm at scale. However, this essay has also shown that LLMs have significant and varied applications to SE. From code completion to API synthesis, LLMs are already reshaping the role of the human SWE, equipping them with new, evolving and powerful tools.

Given the world-changing potential of software, the true significance of this intersection lies in software's capacity to amplify impact far beyond the domain of SE itself. By reducing the unit cost of SE, increasing productivity and democratising a deeply technical skillset, LLMs applied to SE have the potential to change the world. The question, therefore, is not whether LLMs will transform SE, but to what extent and in which domains that transformation will occur.

References

AlphaEvolve Team, 2025. *AlphaEvolve: A Gemini-powered coding agent for designing advanced algorithms* [Online]. Available from: `https://deepmind.google/discover/blog/alphaevolve-a-gemini-powered-coding-agent-for-designing-advanced-algorithms/`.

Anthropic, 2025. *Anthropic economic index: AI's impact on software development* [Online]. Available from: `https://www.anthropic.com/research/impact-software-development`.

Baker & Hostetler LLP, 2025. *Case tracker: artificial intelligence, copyrights and class actions* [Online]. Accessed 22 Jun 2025. Available from: `https://www.bakerlaw.com/services/artificial-intelligence-ai/case-tracker-artificial-intelligence-copyrights-and-class-actions/`.

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

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33, pp.1877–1901.

Buyl, M., Rogiers, A., Noels, S., Bied, G., Dominguez-Catena, I., Heiter, E., Johary, I., Mara, A.-C., Romero, R., Lijffijt, J., et al., 2024. *Large language models reflect the ideology of their creators.* Unpublished. arXiv: `2410.18417`.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.D.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al., 2021. *Evaluating large language models trained on code.* Unpublished. arXiv: `2107.03374`.

Cognition AI, 2024. *SWE-bench technical report* [Online]. Available from: `https://cognition.ai/blog/swe-bench-technical-report`.

Cohen, A., Hoover, A., and Schoenbach, G., 2024. *Watermarking language models for many adaptive users.* Unpublished. arXiv: `2405.11109`.

Cooper, A.F., Gokaslan, A., Cyphert, A.B., De Sa, C., Lemley, M.A., Ho, D.E., and Liang, P., 2025. *Extracting memorized pieces of (copyrighted) books from open-weight language models*. Unpublished. arXiv: 2505.12546.

CVE Program, 2025. *CVE-2025-32711: Vulnerability record* [Online]. Available from: https://www.cve.org/CVERecord?id=CVE-2025-32711.

Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., and Lou, Y., 2023. *Classeval: A manually-crafted benchmark for evaluating llms on class-level code generation*. Unpublished. arXiv: 2308.01861.

Friedman, M., 1970. The social responsibility of business is to increase its profits. *The new york times magazine* [Online]. Available from: https://www.nytimes.com/1970/09/13/archives/a-friedman-doctrine-the-social-responsibility-of-business-is-to.html.

GitHub, 2024. *Quantifying GitHub copilot's impact on developer productivity and happiness* [Online]. Available from: https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/.

Gomes, L., da Silva Torres, R., and Côrtes, M.L., 2023. BERT-and TF-IDF-based feature extraction for long-lived bug prediction in FLOSS: a comparative study. *Information and software technology*, 160, p.107217.

Henrich, J., Heine, S.J., and Norenzayan, A., 2010. The weirdest people in the world? *Behavioral and brain sciences* [Online], 33(2–3), pp.61–83. Available from: https://doi.org/10.1017/S0140525X0999152X.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., and Wang, H., 2024. Large language models for software engineering: A systematic literature review. *Acm transactions on software engineering and methodology*, 33(8), pp.1–79.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I., 2024. *LiveCodeBench: Holistic and contamination free evaluation of large language models for code*. Unpublished. arXiv: 2403.07974.

Jin, M., Shahriar, S., Tufano, M., Shi, X., Lu, S., Sundaresan, N., and Svyatkovskiy, A., 2023. Inferfix: End-to-end program repair with llms. *Proceedings of the 31st ACM joint european software engineering conference and symposium on the foundations of software engineering*, pp.1646–1656.

Meisel, W., 2013. *The software society: Cultural and economic impact* [Online]. Trafford Publishing. Available from: https://books.google.co.uk/books?id=kHQvwkRvVqUC.

Meta AI, 2024. *Introducing Meta Llama 3: The most capable openly available LLM to date* [Online]. Available from: https://ai.meta.com/blog/meta-llama-3/.

OpenAI, 2025. *How your data is used to improve model performance* [Online]. Available from: `https://help.openai.com/en/articles/5722486-how-your-data-is-used-to-improve-model-performance`.

Padmakumar, V. and He, H., 2024. *Does writing with language models reduce content diversity?* [Online]. arXiv: `2309.05196 [cs.CL]`. Available from: `https://arxiv.org/abs/2309.05196`.

Rainie, L., 2025. Close encounters of the AI kind: The increasingly human-like way people are engaging with language models. *Imagining the digital future center.*

Reid, E., 2024. *AI overviews in google search* [Online]. Available from: `https://blog.google/products/search/generative-ai-google-search-may-2024/`.

Russell, S. and Norvig, P., 2022. *Artificial Intelligence: A modern approach.* Pearson Education Limited.

Sachs, J.D., 1999. Twentieth-century political economy: A brief history of global capitalism. *Oxford review of economic policy*, 15(4), pp.90–101.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P.F., 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33, pp.3008–3021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

West, S.M., Whittaker, M., and Crawford, K., 2019. Discriminating systems. *Ai now*, 2019, pp.1–33.

White, C., Dooley, S., Roberts, M., Pal, A., Feuer, B., Jain, S., Shwartz-Ziv, R., Jain, N., Saifullah, K., Dey, S., Shubh-Agrawal, Sandha, S.S., Naidu, S.V., Hegde, C., LeCun, Y., Goldstein, T., Neiswanger, W., and Goldblum, M., 2025. LiveBench: a challenging, contamination-free LLM benchmark. *The thirteenth international conference on learning representations.*

Wilson, B., Hoffman, J., and Morgenstern, J., 2019. Predictive inequity in object detection. *Corr* [Online], abs/1902.11097. arXiv: `1902.11097`. Available from: `http://arxiv.org/abs/1902.11097`.

Yang, Y., Xia, X., Lo, D., and Grundy, J., 2022. A survey on deep learning for software engineering. *Acm computing surveys (csur)*, 54 (10s), pp.1–73.

Zhivich, M. and Cunningham, R.K., 2009. The real cost of software errors. *Ieee security & privacy*, 7(2), pp.87–90.