Building an AI-Native Software Engineering Team: A Stepwise Approach using Multi-Agent Systems

Hariharan Balasubramani

ABSTRACT
The realm of Generative Artificial Intelligence (Gen AI) has propelled human ingenuity to unprecedented heights, promising to revolutionize the field of software engineering. Large Language Models (LLMs) and Generative Pre-trained Transformers are at the forefront of this transformation, reshaping the landscape of Software Engineering. With the integration of multi-agent systems, the evolution of software engineering is poised to accelerate even further. Multiple generative agents interacting with each other can handle not only basic tasks like coding, debugging, and scripting, but also creativity-intensive tasks and other aspects of the software engineering lifecycle such as requirement gathering, software design, project planning, QA testing, and documentation. Human engineers will play a crucial role in providing high-level instructions and making course corrections. The emergence of AI-native firms with AI-driven software engineering teams will lead to significantly reduced turnaround times for ideas to become finished products. This approach will streamline the entire software development process, from requirement gathering and planning to the final product, resulting in faster delivery and lower production and operational costs compared to traditional IT firms. In this paper I will provide empirical evidence for the above claims and a stepwise framework for building such a team.

Keywords
Large Language Models, Artificial Intelligence, Software Engineering, Multi Agent System, Generative AI

1. INTRODUCTION
Just as how we saw the widespread adoption and affordability of cloud computing technology led to the rise of cloud-native firms, we are now witnessing the emergence of AI-native firms. These companies will leverage the power of artificial intelligence to revolutionize business operations and deliver value to customers. As generative AI makes software engineering more accessible, new companies can develop software without the need for maintaining an expensive team of software developers. This trend is creating a new profession, the AI Native engineer, focused on training models and integrating AI-created software for endusers.AI-native firms, free from bureaucratic processes and legacy IT planning and decision strategies, can rapidly build and deploy software using the most efficient frameworks. These firms disrupt industries by significantly reducing time-to-market, measured in weeks instead of months, and compelling existing firms to innovate or be acquired [1]. While AI-driven software engineering may initially lag human-driven development, its speed and cost-effectiveness will make it appealing to many customers, provided it maintains reliability, security, and core value.

1.1 Software Engineering
Software Engineering (SE) is a branch of computer science which focuses on systematically and predictably designing, developing, testing, and maintaining software systems [4]. In today's landscape, where the software industry serves as the backbone of numerous other industries, software engineering (SE) assumes a pivotal position in modern society. Its primary responsibility lies in ensuring that systematic, reliable, and efficient software systems are built [5]. Apart from this software engineering can be termed as a field that requires a variety of different skillsets for an engineer to thrive.

1.2 Large Language Models in SE
Large Language Models (LLMs) are deep learning models that are pre-trained on large corpuses. They are built upon transformers, which are neural network architectures consisting of an encoder and a decoder. These enable the extraction of semantic relationships from sequences of texts and understand the interrelationships between them. Advanced LLMs like BERT[7], GPT-4[8] have significantly enhanced performances across a wide range of NLP based tasks. More and more coding centric LLMs are being developed such as codellama, deepseek coder, phi and many more.[5]. There are three types of LLMs for coding specific tasks the encoder-only, decoder-only and encode-decoder based LLMs. These LLMs are specifically trained on large corpuses of code related data. Some of the most common tasks that LLMs perform in SE are code generation, code completion, code search, API synthesis, comment generation, etc. [9]

Below is an example of different context or corpuses can be used to train or used in conjunction with a model to perform a code translation task.

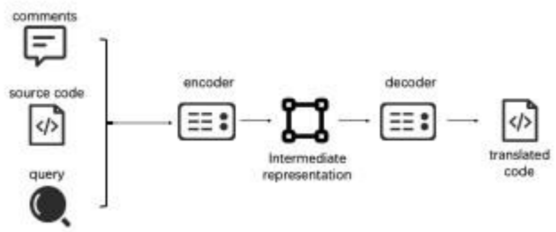

Figure 1: Example of a code translation task by an LLM[5]

1.3 Multi-Agent Systems (MAS)
This system refers to the use of multiple autonomous agents to solve complex problems. Each of these agents have their own goals, knowledge, and reasoning. In our context these agents can represent various entities like coder, tester, developer, etc. These individual agents interact to achieve a collective objective. This is commonly referred to as Multi-Agent System (MAS)[6].The primary advantage of MAS is their ability to capture the complexity and dynamics of a software development process, which are often characterized by uncertainty and non-linearity.[10]

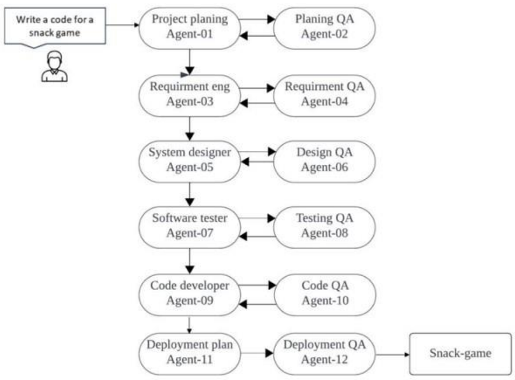

Figure 2 : Multiple Agents working for generating the code for a simple game [6]

1.4 Small Language Models

For rapid prototyping of AI models to be feasible the basic requirement is to be able to run the training from a developer's laptop. LLMs with 7B to 1B parameters can fit and run in a good configuration system. But running complex models with a lot of parameters on a small machine is virtually impossible. A breakthrough to solve this problem is tackled by running small models with textbook quality data, which are small, use lesser compute but are still performant. For software development this opens a performant and environmentally sustainable option.

2. LITERATURE REVIEW

The blog post by Omar Ansari[1] was a huge inspiration for my paper. It led me to understand why addressing this is a need for the hour. In their work, Quanjun et al have summarized the usage of LLMs in Software Engineering tasks. Their paper also offers insight about how different LLMs are used for source code translation, syntax validation, pre-training, and other tasks downstream. This covers the holistic life cycle of different SE tasks in different phases. Zeeshan et al in [6] have covered the role of autonomous agents in SE tasks. This was the foundation of my thought process around how a firm will approach or handle building an AI-Native team. It not just talks about how different tasks in SE are addressed by these agents but also describes how this improves the accuracy and quality of the code that is generated.

3. PROPOSED FRAMEWORK

Large IT firms will soon be feeling the pressure caused by reduced time to market and low operation cost products created by AI driven firms. To be prepared for this, many firms are encouraging their engineers to experiment in a safe way with AI and how it can be integrated into different parts of their SDLC and product life cycle. Companies that are pro-actively engaging in this will have a head start in the race against their competitions and will also be more resilient.

The journey of each firm will be different as each of them are unique in their own way. Each of them might face specific challenges brought in by the complexity of their systems such as using languages / platforms/tools which are incompatible with AI. They might have complex architectures which are unable to accommodate the new changes as a result of unique customizations, technical debts, etc. Even simple things like documentation, how to structure a code commit and style checks need to change to accommodate AI integration.

The first step in bridging this gap is to understand the current state and take the first step as close to the target state as possible. Then there needs to be constant course corrections to make sure we progress in the right direction. Starting with extensions for IDEs and performing AI assisted development is a great first step that many companies have started doing. However, it is not sufficient. Let us examine a detailed stepwise framework for such a transition.

Phase 1 : Evaluation and Strategizing

This is the first and foremost step in any major change. Assessment and Planning are the core of the entire process.

a. Evaluate and Plan

Evaluate the software development processes while trying to understand the core tasks or responsibilities of each engineer like planning, coding, testing and documentation. This evaluation helps identify the extra services needed for teams to integrate an AI Software Engineer into the existing software delivery workflow.

b. IT Infrastructure and Environment Evaluation:

Identify potential challenges for AI agents by thoroughly examining the deployment and management process. This includes analyzing authentication management and the deployment steps that are currently performed manually. Streamlining these steps will be crucial to ensure efficiency and scalability. Additionally, addressing these issues can significantly reduce operational overhead and improve the overall effectiveness of AI agent deployment.

c. Adopt and Assess:

Identify early adopters and applications that would benefit the most from AI integration, and evaluate the potential impact. Use scoring and metrics to perform a comprehensive cost-benefit analysis. By targeting early adopters, you can effectively showcase the advantages of AI integration. Additionally, this analysis will help in prioritizing applications based on their potential return on investment and strategic value.

d. Project Planning:

Create a well-defined plan with timelines and a detailed roadmap with long term and short-term goals.

e. Prototype and test:

Perform a POC driven testing of different AI technologies. This will enable a quick feedback loop and reap all benefits of a test-driven development. It is highly advisable to form a highly efficient, close-knit cohort consisting of software engineering leader, an architect with in-depth production and deployment knowledge and a lead engineer. This Elite team will plan and oversee the smooth transitioning.

Phase 2: Creating an internal AI Consultant Division

Each individual team/developer experimenting with AI in their daily activities is important. However, establishing this internal division is very critical. This division should consist of the following core teams.

a. Central AI Group:

The main responsibility of this group is to stay up to speed with advancements related to AI. This team builds, customizes, and maintains custom AI models for the firm. They are also responsible for establishing data contracts for sharing data with the model, fine-tuning the model, and building guard rails around misuse of the system.

b. Core Data Platforms:

This team takes care of the firm's end to end data needs. From data gathering to ensuring quality and security. This data will be used to train the AI models hence accuracy, relevance and consistency matters a lot.

c. Integration and Testing Team:

This team takes care of integrating the build models into various systems of the firm and evaluate the outcome. This is very critical as this will increase the wider adoption and integration.

d. Ethics and Compliance:

Ensures the ethical development and deployment of AI models. It also takes care of building the guard rails for compliance concerns.

The milestone to mark the end of this phase will be creating a fully functional SDE (Software Development Engineer) Agent and testing it with an actual scrum/team.

Phase 3: Introducing the new team member into the scrum Mr.AI-SDE

This phase involves the on-field testing where the Agent is introduced into an actual scrum for a team with the supervision of the Lead Engineer. The agent can start with some simple tasks such as making a code pull, documentation, resolving a technical debt. This is to check how good the AI-SDE is in comparison to a real human developer.

agent's name	Responsibility	Interacts with	Applicable LLM
User Proxy	Takes inputs from the Lead Engineer, either mid-steps or after full interactive tasks are complete	Lead Engineer (human)Planner agent	Mistral (OSS)
Planner	Creates an execution detailed step by step plan (think what humans do for 1 point user stories)	Critic RAG SWE QA	Orca2 (OSS) [good at reasoning]
Critic	Evaluates and sharpens the plan created by the planner	Planner	Orca2
RAG	Retrieval augmented generator pulls data from different sources such as an internal or external website, or documents and provides it back with context and references	PlannerSWE	Phi-2Deepseek coderCodellamaGPT4
SWE	Software engineer: Builds the software to spec receiving relevant context and knowledge from RAG, sends to executor to run, fixes defects identified in run-time, or issues identified by code reviewer, and provides results bcak to Planner	RAG+1Executor	Custom internal model built on pre-trained coder such as Phi-2

Figure 3: The Agent can internally have the following agents and responsibilities [1]

An interesting thing to note here is this AI agent is a combination of multiple agents each one specializing in a domain. This testing allows the Lead Engineer to finalize what tuning is needed for each of these individual agents.

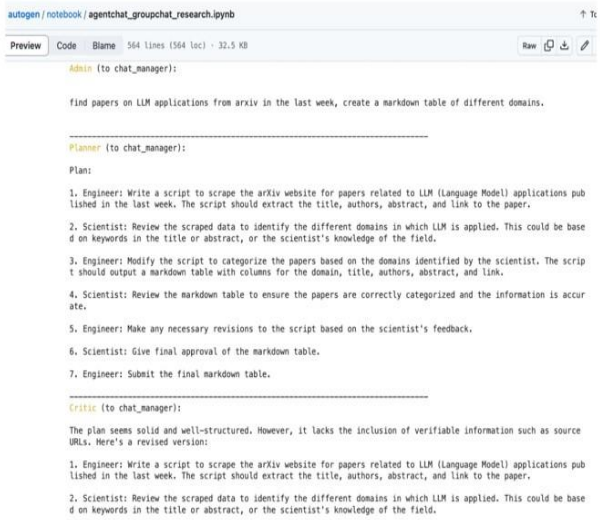

Figure 4: Multi-Agent examples showing how mini agents talk to and critique each other to produce a superior final output.[12]

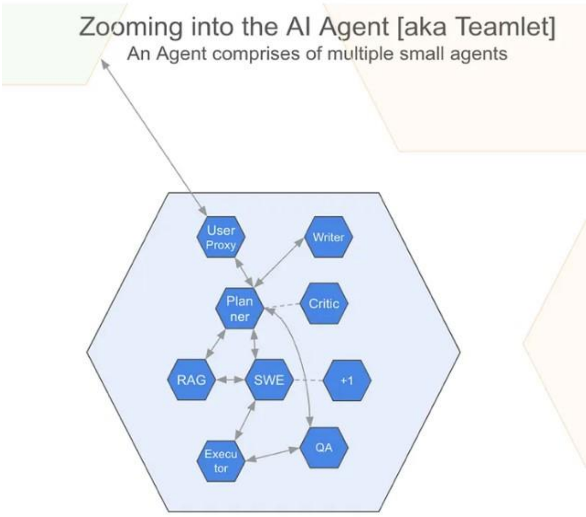

Figure 5: Zooming into an AI Agent [1]

The milestone for completion of this phase can be at least each division/team having one such agent and then being able to successfully pick up a task from the scrum and complete it while also updating the project management system and maintaining code quality.

Phase 4: Introducing semi-AI Native teams into each domain.

This phase marks creation of teams completely made of AI agents. The teams should now be able to complete a whole epic by themselves, demanding a much higher level of communication with other agents and humans. Agents can have different roles like SDE, Product Manager, Scrum Master, etc. each specializing in a specific task/domain.

While AI is becoming better in coding, communication, and planning, it currently struggles with non-functional requirements like performance and scalability. Human intervention is needed to ensure these benchmarks are met.

The milestone to mark this phase complete can be handling complete epics while using human supervision for tasks requiring complex critical thinking and dealing with nonfunctional requirements.

This phase marks introducing AI agents into other teams drawing from the experience gathered in the last phase. Senior engineers can now assume more responsibilities since AI agents take care of implementations. An end-to-end QA team, comprising both AI and human engineers, is crucial. This team ensures the end-to-end experience is verified and integration-tested before release, including performance testing. This phase will see a clear reduction in time to develop for features since the AI agents are more widely adopted into teams.

Final Phase: Decentralizing AI Shared Services

With the scaling out of AI teams, human SDEs take up more high-level responsibilities, supervising multiple AI agents / teams. Their focus will be to review and approve code changes and documentation of the AI SDEs. While the AI agents can handle most tasks, they will always need human engineering teams to act as a safety net, making sure nonfunctional requirements are catered.

Decentralizing AI Shared Services becomes important as technology and products mature. Each domain should build its own muscle to customize AI models, curate training data, and maintain custom integrations. The AI consultant Division transitions into an AI Standards Group (ASG), focusing on horizontally required services and maintaining standardized integrations. The ASG also ensures code quality and security, potentially using sentinel software to track and report on the overall security posture centrally.

4. CONCLUSION

To sum up, the implementation of AI-driven software development and engineering (SDE) heralds a paradigm shift that is fraught with opportunities and difficulties. Human developers still have a significant advantage in creativity, problem-solving, and subtle comprehension even though AI is excellent at repetitive activities and can increase efficiency. Organizations should carefully incorporate AI into their teams to maximize benefits while minimizing dangers. This involves maintaining human oversight over crucial jobs like code quality and non-functional requirements. Additionally, to solve ethical, social, and dependability problems, it is imperative to establish a balance between AI and human expertise. Engineers should concentrate on developing unique abilities like creativity, problem-solving, and communication as the industry develops since they are talents that will be useful in the AI-powered SDE environment. Eventually, businesses that accept and adjust to these changes will thrive or at least have a head start when compared with their peers.

Metric	Factors
Code Quality	- Number of bugs per 1000 lines of code (LOC)
	- Frequency of critical errors
Development Efficiency	- Development time reduction (days or hours saved)
	- Percentage of tasks automated by AI
Testing and Debugging	- Test coverage percentage
	- Time spent on testing and debugging
	- Number of test cases generated and executed by AI
Productivity	- Number of releases per year
	- Feature delivery rate
	- Time to market for new features
Collaboration	- Frequency and quality of human-AI interactions
	- Efficiency in problem-solving (measured by time or successful resolutions)
Employee Satisfaction	- Employee satisfaction scores related to AI integration
	- Surveys on the perceived usefulness of AI tools
Cost-Benefit Analysis	- Cost savings from AI integration (e.g., reduced labor costs, increased efficiency)
	- Return on investment (ROI) for AI projects
Training and Adaptation	- Time required for team members to adapt to AI tools
	- Number of training sessions conducted and participation rates
Innovation and Creativity	- Number of innovative solutions or ideas generated
	- Improvement in creative problem-solving scores
Ethical and Social Impact	- Instances of ethical or social issues arising from AI use
	- Measures taken to address ethical concerns
Reliability and Maintenance	- Downtime or failures attributable to AI
	- Frequency and effectiveness of AI maintenance activities
Compliance and Security	- Compliance with industry standards and regulations
	- Number of security incidents related to AI

Figure 6: Metrics and Factors for evaluation

Expanding on this, it is essential to conduct thorough assessments of AI tools before integration to ensure they align with the organization's goals and values. Implementing continuous training programs for engineers to familiarize them with AI technologies can significantly enhance the effectiveness of these tools. Moreover, fostering a collaborative culture where AI and human expertise are seen as complementary can drive innovation and productivity. For instance, AI can handle large-scale data analysis and testing, allowing human developers to focus on strategic planning and complex design problems. Regularly reviewing the outcomes of AI integration through scientific tables and metrics, , can provide valuable insights into performance improvements and areas needing adjustment. By leveraging AI's strengths while safeguarding human-centric skills, organizations can navigate this transformative era with confidence and foresight.

5. ACKNOWLEDGEMENT

I would like to extend my sincere thanks to my mentors and colleagues for their unwavering encouragement and inspiration in pursuing this topic. Their guidance and support have been invaluable throughout this journey. Additionally, Omar Ansari’s blog provided brilliant insights that propelled me to write this paper. His detailed analysis and innovative ideas offered a fresh perspective that was instrumental in shaping my research. I am deeply grateful for all the knowledge and motivation I have received from these remarkable individuals.

6. REFERENCE

[1] Towards Data Science, 2020. *Designing AI-Driven Software Engineering Teams*. [online] <https://towardsdatascience.com/designing-ai-drivensoftware-engineering-teams-8afd8de13f1a> [Accessed 21 May 2024].

[2] Saklamaeva, V. and Pavlič, L., 2023. The Potential of AIDriven Assistants in Scaled Agile Software Development. *Applied Sciences*, *14*(1), p.319.

[3] Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J. and Wang, H., 2023. Large Language Models for Software Engineering: A Systematic Literature Review. *arXiv preprint*, arXiv:2308.10620 [cs.SE].

[4] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, "Systematic review in software engineering," System engineering and computer science department COPPE/UFRJ, Technical Report ES, vol. 679, no. 05, p. 45, 2005.

[5] A Survey on Large Language Models for Software Engineering Quanjun Zhang, Chunrong Fang, Yang Xie, Yaxin Zhang, Yun Yang, Weisong Sun, Shengcheng Yu, Zhenyu Chen arXiv:2312.15223 [cs.SE]

[6] Autonomous Agents in Software Development: A Vision Paper Zeeshan Rasheed, Muhammad Waseem, Kai-Kristian Kemell, Wang Xiaofeng, Anh Nguyen Duc, Kari Systä, Pekka Abrahamsson arXiv:2311.18440 [cs.SE]

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pretraining of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[8] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[9] Xinyun Chen, Chang Liu, and Dawn Song. 2017. Towards synthesizing complex programs from input-output examples. arXiv preprint arXiv:1706.01284 (2017).

[10] Wei Ma, Shangqing Liu, Wenhan Wang, Qiang Hu, Ye Liu, Cen Zhang, Liming Nie, and Yang Liu. 2023. The Scope of ChatGPT in Software Engineering: A Thorough Investigation. arXiv preprint arXiv:2305.12138 (2023).

[11] Feng et al , Investigating Code Generation Performance of Chat- GPT with Crowdsourcing Social Data

[12] Cañas, J.J., 2022. AI and ethics when human beings collaborate with AI agents. *Frontiers in psychology*, *13*, p.836650.

[13] Belloni, A., Berger, A., Boissier, O., Bonnet, G., Bourgne, G., Chardel, P.A., Cotton, J.P., Evreux, N., Ganascia, J.G., Jaillon, P. and Mermet, B., 2015, April. Dealing with ethical conflicts in autonomous agents and multi-agent systems. In *Workshops at the twenty-ninth AAAI conference on artificial intelligence*.