Analysing the Role of Generative AI in Software Engineering - Results from an MLR

Tuomas Bazzan[1], Benjamin Olojo[1], Przemysław Majda[1], Thomas Kelly[1], Murat Yilmaz[2] [0000-0002-2446-3224], Gerard Marks[3], and Paul M. Clarke[1,4] [0000-0002-4487-627X].

[1] School of Computing, Dublin City University, Ireland
{tuomas.bazzan2, benjamin.olojo2, przemyslaw.majda2, thomas.kelly46}@mail.dcu.ie
[2] Department of Computer Engineering, Gazi University, Ankara, Turkey
my@gazi.edu.tr
[3] Faculty of Engineering and Computing, Dublin City University
gerard.marks@dcu.ie
[4] Lero, the Science Foundation Ireland Research Centre for Software
paul.m.clarke@dcu.ie

Abstract.
Generative Artificial Intelligence (GenAI) has become a practical tool that exhibits the potential to revolutionize numerous industries through publicly available systems with simple yet effective interfaces. This paper outlines the findings of research conducted in a multivocal literature review (MLR) with the aim of exploring the impact of GenAI in software engineering, with a focus on the fundamental aspects, use cases, benefits, and risks associated with contemporary GenAI models leveraged in key industries and practices. Key findings indicate that GenAI is adopted in software engineering, with various reported benefits in areas including requirement engineering, estimation and testing. However, there are also some risks associated with GenAI-based Software Engineering, such as in the context of generated data consistency and accuracy (sometimes referred to as the Hallucination problem), plagiarism, bias, and security. GenAI-assisted software engineering is becoming more mainstream, but resolving all the associated issues is going to take some time.

Keywords: Generative AI, Large Language Models, Software Engineering, Risks, Benefits.

1 Introduction

In recent years, Generative Artificial Intelligence (GenAI) has emerged as a groundbreaking force in our technological landscape, with the potential to orchestrate a profound transformation across diverse industries and propel us into an era of unprecedented innovation and creativity. Driven by advanced algorithms and immense computational capabilities, GenAI possesses the ability to redefine creativity and productivity,

generating novel content that spans text, images, videos or programming code in response to instructions, or prompts, entered by the respective user [1].

When ChatGPT was launched in late 2022, it was quickly established as a seminal technology and awakened the world to the transformative potential of GenAI [1]. The technology underpinning the revolutionary chatbot represents one of the biggest step changes in the history of artificial intelligence—rather than simply analysing or classifying existing data, GenAI is able to create entirely new and diverse material [1].

From ChatGPT to DALL-E, the latest class of GenAI applications has emerged from foundation models, which are complex machine learning systems trained on vast quantities of data (text, images, audio or a mix of data types) on a massive scale [1]. As depicted in Figure 1, by leveraging these foundation models GenAI models can handle varied tasks involving different input and output types, such as converting text into images using the DALLE-2 model, text into audio using the AudioLM model or text into code by the Codex model [2].

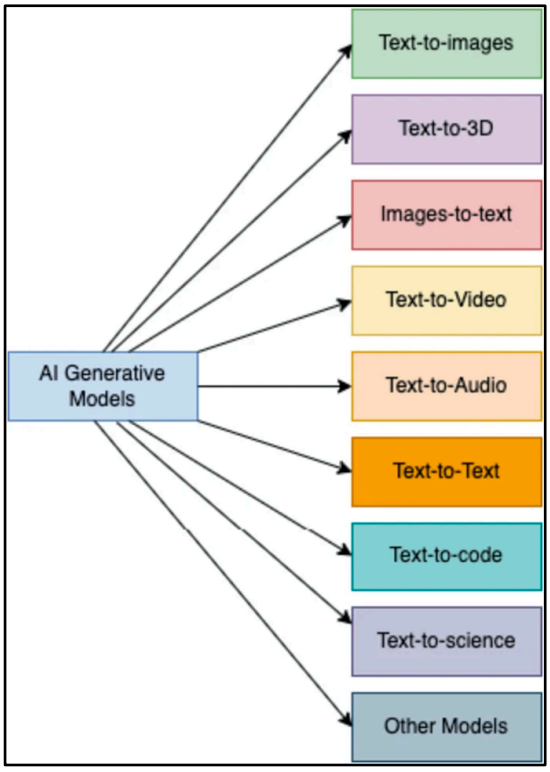

Figure 1 - A taxonomy depicting the classification of popular generative AI models based on their input and the format of the output they generate [2]

The power of these GenAI systems lies not only in their size and computational capabilities but also in the fact that they can be adapted quickly for a wide range of downstream tasks without the need for task-specific training, making them suitable for several domains [1]. Through ever-increasing utility and adoption, GenAI consistently challenges the previous assumption that artistic, creative tasks such as writing poems, creating software, designing fashion, and composing music could only be performed by humans [3].

This research aims to examine the current and potential impact of GenAI on the software engineering industry through four key themes: technological foundations, use cases, benefits and risks. These themes comprise the following research questions considered within this paper:

- **RQ1:** What is Generative AI?
- **RQ2:** How is Generative AI used in Software Engineering?
- **RQ3:** What are the benefits associated with using Generative AI in Software Engineering?
- **RQ4:** What are the risks associated with using Generative AI in Software Engineering?

The paper follows an organised structure: Section 2 outlines the research methodology, while Section 3 provides an analysis of the related literature through four aforementioned research questions. Section 4 briefly discusses the known limitations of research, with Section 5 indicating directions for future research. Section 6 presents concluding remarks.

2 Research Methodology

2.1 Methodology

This research paper was written as a Multivocal Literature Review (MLR) with the result that it includes white literature (peer reviewed papers) and grey literature (blogs, articles, news reports, etc.). Platforms such as Google Scholar and IEEE Xplore were used to search and identify academic literature on our chosen topic, as well as standard search engines like Google Search to find relevant grey literature.

2.2 Search Strings

Initial search query strings, such as “Generative AI in Software Engineering”, were deployed to identify central areas of interest. Refined search strings were then elaborated and incorporating Boolean logic, including, “Foundation Models”, “Large Language Models” AND “Training”, “Generative AI” AND “Software Engineering”, “Generative AI” AND “Benefits” OR “Gains”, “Generative AI” AND “Software

Engineering" AND "Risks", "Generative AI in Software Engineering" AND "Hallucination", "Intrinsic Hallucination".

2.3 Inclusion/ Exclusion Criteria

As part of our inclusion/exclusion criteria for obtaining relevant and reliable material, we opted to restrict our literature selection to mainly academic works published after 2019 using Google Scholar. Given the rapid evolution of the generative AI field, we highlighted that it was imperative for our research to remain current and pertinent. Once we compiled relevant white literature material using our search strings, we proceeded to analyse the abstracts, introductions, and conclusions of each paper to assess their relevance and reliability. We then reviewed the selected literature and analysed key points while considering the diverse perspectives from different papers and their listed citations. Researching grey literature required a more cautious approach that considered only those that were deemed to be entirely relevant and reliable sources. Exclusion criteria included non-relevant papers, material that wasn't accessible in full format or in English, and those whose origin or authenticity could not be established. Finally, we constructed a literature pool containing information on material containing various insights systematically obtained through our MLR.

3 Analysis

3.1 RQ1: What is Generative AI?

To effectively explore the use cases, benefits, and risks of GenAI in software engineering, we must first establish a definition of GenAI and examine its foundational aspects, capabilities and ongoing evolution.

Definition of Generative AI
In contrast to discriminative AI models, which focus on modelling input data decision boundaries in tasks such as classification, regression, or clustering, GenAI models excel at the probabilistic generation of new data such as text, images, audio, and videos [4]. GenAI is underpinned by foundation models consisting of deep artificial neural networks with billions of parameters trained on massive unannotated datasets [5][6]. These datasets can consist of content retrieved from diverse sources such as books, articles, websites, and social media posts, among other sources [5]. Foundation models can serve as the backbone for various GenAI tasks after extensive training on large and diverse corpora of data, allowing them to acquire comprehensive and expressive representations of complex data, such as natural language, digital images and audio [8].

The most widely known foundation models are large language models (LLMs), such as ChatGPT [5]. Other well-known categories of foundation models include multimodal and visual language models (VLMs). Multimodal models can generate multiple types of data through understanding the connections and interactions between them [7].

VLMs learn the visual representations of objects from large-scale image datasets and utilise these learned representations for downstream computer vision or vision-language tasks [8]. GenAI models, being built upon these foundation models, are then capable of multi-tasking and performing complex tasks based on input prompts including summarisation, question answering, text and image classification, image generation and code generation [8].

Modern GenAI models, such as those identified in Figure 2, work by leveraging the statistical capabilities of deep generative models (DGMs) and deep learning techniques to model high-dimensional probability distributions, enabling them to produce outputs that resemble real-world data [8]. Deep learning refers to an advanced subset of machine learning that leverages artificial neural networks, with multiple hidden layers in a nested architecture, to model complex data representations and automatically detect correlations and patterns in large datasets [6][7]. Training DGMs is typically conducted through semi-supervised learning, a combination of learning techniques using a small amount of labelled data in a supervised approach, followed by extensive unlabelled data in an unsupervised approach [8].

Evolution of Generative AI

GenAI models within the domain of artificial intelligence date back to the 1950s, with the development of Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs), which generated sequential data such as speech and time series data [11]. However, it wasn't until the emergence of deep learning that GenAI models saw significant improvements in performance [11].

In the field of natural language processing (NLP), a traditional method to generate sentences was to learn word distributions using N-gram language modelling and then search for the predicted optimal sequence [11]. However, this method couldn't effectively adapt to long sentences, which were addressed by the introduction of recurrent neural networks (RNNs) for language modelling tasks, allowing for modelling relatively longer dependencies in the text [11]. This was followed by the development of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which utilised a gating mechanism to control memory management during model training [8].

Within the field of computer vision, before the advent of deep learning-based methods, traditional image generation algorithms used techniques such as texture synthesis and texture mapping. These algorithms were based on hand-designed features and limited in their ability to create complex and diverse images [11].

The advancement of generative models in various domains has followed different paths but the arrival of the transformer architecture served as a key factor in the development of GenAI models [8]. Introduced by Vaswani et al. for NLP tasks in 2017 to solve the limitations of traditional models such as RNNs in handling variable-length sequences and context awareness [8], the transformer was later applied in Computer Vision and

then became the dominant backbone for many generative models in various domains and many state-of-the-art models such as GPT-3, DALL-E-2, Codex, and Gopher [11].

Category	Tool	Company	Country
Chatbots	ChatGPT	OpenAI (Microsoft)	USA
	Bard	Google	USA
	Bing AI	Microsoft	USA
	HuggingChat	Hugging Face	USA
	Jasper AI	Jasper	USA
	ChatFlash	neuroflash	Germany
Large language models	GPT-4, GPT-3.5	OpenAI (Microsoft)	USA
	Claude 2	Anthropic	USA
	Luminous	Aleph Alpha	Germany
	cohere com-mand	Cohere	Canada
	PalM2	Google	USA
	LLaMA	Meta	USA
Code generation	AlphaCode	DeepMind (Google)	USA
	GitHub Copilot	GitHub(Microsoft)	USA
	Tabnine	Tabnine	Israel
	Open AI Codex	Open AI	USA
	Codebase	MutableAI	USA
	Replit AI	Replit	USA
	Codacy	Codacy	Portugal
Image/Videos	DALL-E2	OpenAI (Microsoft)	USA
	Imagen	Google	USA
	Stable Diffusion	Stability AI	UK
	Synthesia	Synthesia	UK
	Midjourney	Midjourney	USA
	Openart	OpenArt	USA

Figure 2 - A table outlining the contemporary competitive landscape of GenAI [1].

The transformer architecture utilises encoder and decoder blocks with a self-attention mechanism that allows the model to attend to different parts in an input sequence [11]. With the aim of solving sequence-to-sequence problems, the encoder takes in an input sequence and generates intermediate hidden representations, while the decoder takes in the hidden representation and generates a relevant output sequence [11].

In the field of NLP, many prominent LLMs such as BERT and GPT models adopt the transformer architecture as their foundational building block, offering advantages over previous building blocks such as LSTM and GRU [11]. Similarly in Computer Vision, the Vision Transformer (ViT) and Swin Transformer later took the theory applied by Vaswani et al. even further by combining the transformer architecture with visual components, allowing it to be applied to image based downstream tasks [11]. In addition to the improvement that transformers have brought to individual modalities, this intersection also enabled models from different domains to be fused together for multimodal AI tasks. [11]

Large Language Models

A Large Language Model (LLM) is a deep neural network model which has been trained on large amounts of textual data, such as books, code, articles, and websites, to learn the underlying patterns and relationships in specific languages with the aim of building an internal representation of the ingested data through analysis of language structure, rules and patterns [1]. Through this, the large language model is able to generate coherent and novel content such as grammatically correct sentences and paragraphs that mimic those generated by humans in its training data, or syntactically correct code snippets and documentation also based on training data from large code repositories [1].

LLMs have emerged as cutting-edge artificial intelligence systems, playing a pivotal role in the progression of natural language processing capabilities which have evolved initially from statistical to neural language modelling and eventually from pre-trained language models (PLMs) to LLMs [12]. Pre-training language models involve providing them with universal language knowledge [13], while meta-training aims to align the model's behaviour to the user's intentions. In this case, the user's intention includes both explicit intentions, like following instructions, and implicit intentions, like maintaining truthfulness and avoiding bias, toxicity, or any harmful behaviour [14]. Large language models can then be considered a special class of pre-trained language models developed by scaling the model size, pre-training corpora and computation resources [12].

In addition to improved generalisation and domain adaptation when prompted, LLMs also appear to have what are known as emergent abilities, which refer to unpredictable tasks performed by an algorithm, that weren't explicitly programmed into it [15]. For example, the ability of a LLM to capture the non-specified semantic relations between words based on their vector representation in multidimensional space can be considered an emergence phenomenon [15]. These abilities are known to be acquired by LLMs due to their enormous parametric scale even when the pre-trained LLMs are not trained specifically to possess these attributes [12]. Such abilities have led LLMs widely adopted in diverse settings including, multimodal models, robotics, tool manipulation, question and answering, autonomous agents, etc. Various improvements have also been

suggested in these areas either by domain or task-specific training and better prompting tailored to the intended task and LLM being utilised [12].

3.2 RQ2: How is Generative AI used in Software Engineering?

LLMs excel at processing complex inputs and information to produce relevant new content, features that heavily lend themselves to Software Development. The technology's direct impact on industry productivity is estimated at 20-45% of the current yearly expenditure. This figure mainly represents the reduction in time spent performing certain tasks (code drafts, correction and refactoring, root-cause analysis, generating new system designs) when using GenAI assistance [16].

With such a large economic impact, more and more software departments are incorporating the technology into their workflow, with one survey showing 77% business leaders (across multiple industries) ranking GenAI as impactful emerging technology they will use, and 56% of leaders voting IT/Tech and Operations as the highest priority functions for adoption of the technology [17].

Generative AI in Software Engineering
GenAI has been recognised for having the potential to revolutionise the Software Development Life Cycle (SDLC), with propositions even being made for a new model of SDLC called GAASD (GenAI assisted Software Development) [18]. This new model would incorporate GenAI in all stages of the life cycle, from the Planning and Requirements Gathering stage to the Maintenance phase. As companies shift their focus and resources to GenAI assisted development, GAASD might soon be the next paradigm shift comparable to Agile development in the 2010's. Mass adoption of a new SDLC model is not something that happens overnight, but GenAI is already being used in various stages of a traditional SDLC.

Current Applications
The impact of GenAI is already apparent across the industry landscape, as over 20,000 organisations are already using Github's Copilot [19], a GenAI based coding assistant that integrates directly into the developers' IDE where it provides auto-complete code suggestions and a chat interface for more complex queries. Dr. Ozkaya, technical director at the SEI, identifies 5 main areas that GenAI LLMs can be used in software development [20]:

- *Specification generation:* LLMs can access entire corpora of domain specific regulatory documents and specifications, and excel at processing textual data. Using this information paired with project specific requirements, a LLM could create a comprehensive list of requirements for a software application that would only need to be reviewed by the product manager. In an online Q&A session [21], Douglas Schmidt of the SEI highlights this functionality when asked about how GenAI will be used in the US Department of Defense.

- *Just-in-time developer feedback:* Using a GenAI coding assistant, developers can get instant feedback at any time without the need for a code review or assessment by a peer or superior. Github's Copilot can provide not only corrections and improvements but also suggestions for future steps developers can take, with developers reporting significant improvements in coding efficiency, and feeling fulfilment at work [19].

- *Improved testing:* Another major usage of coding assistants is in the form of automatically generated unit tests. Manual testing has several drawbacks [22] which GenAI coding tools can help mitigate. In a 2024 virtual roundtable [23] software testing expert Paul Gerrard says "I envision that testing and QA will become more focused on the end-user experience and less focused on code-centric activities, such as writing unit tests, as GenAI technologies continue to mature," when asked how GenAI tools will integrate into the software engineering process, reflecting the industry trend of automating less cognitively demanding coding tasks with GenAI.

- *Documentation:* GenAI LLMs can improve documentation in several ways in software development. According to one IEEE journal [24], GenAI can be used to:
    - Create comprehensive documentation as it generates code,
    - Summarise existing documentation,
    - Identify missing requirements or inconsistencies in requirements documentation.

  This can streamline the documentation process and improve the quality by keeping the documentation reflective of the codebase and requirements.

- *Language translation:* Many legacy software systems are built using outdated and obsolete languages, which makes maintenance and integration a time-consuming task that requires software engineers who are proficient in both the legacy and modern languages. In the online Q&A session [21], Douglas Schmidt, former chief technical officer of the Software Engineering Institute (SEI), talks about how LLMs can be trained and fine-tunes to understand languages that are used in the Department of Defence legacy repositories (e.g. Ada, JOVIAL), to maintain and aid in the evolution and comprehension of codebases.

Novel Applications and Trends

In the "Introducing GPTs" November 2023 update, OpenAI's ChatGPT started allowing users to create custom GPTs complete with API calls and instructions, providing a chat interface that can execute custom actions [25]. These actions can enable a GPT to perform tasks outside of the confines of ChatGPT, allowing it to do things such as access external data, connect to databases and send HTTP requests with in-built user authentication. Combined with the Team workspace services that ChatGPT offers, software teams could be able to collaborate in an entirely novel way, letting them interface

with internal and external APIs through plain language prompts. However, as a new feature, there is limited data to support its practicality in real terms.

Tools and workspaces like these are shaping a rapidly changing future in software engineering, potentially altering the fundamental ways in which software engineers approach their jobs. In the previously mentioned IEEE Journal, Ebert and Louridas speculate that within 3 years most software companies will have an AI-augmented development and testing strategy, which will lead to developers requiring new competencies [24]. Previously niche proficiencies like prompt engineering, improving automatically generated software and feeding learning engines may become more valued as the role of software engineer evolves.

Prompt engineering is one of the main emerging areas of interest as GenAI tools become increasingly popular. Prompt engineering is the process of creating and refining detailed instructions (prompts) for a GenAI model, to receive high quality and relevant output [26]. This process requires creativity and trial and error in order to guide the AI system to the best possible response it can create. Prompt engineering may redefine certain problems in software engineering, where "programming tasks are expressed as prompts that guide the behaviour of AI models, thereby encouraging the exploration of creative and innovative strategies over applying traditional programming methods and tools" [27]. This shift in methodology creates new opportunities, as 'prompt engineer' is becoming a standalone job title, as seen in Figure 3.

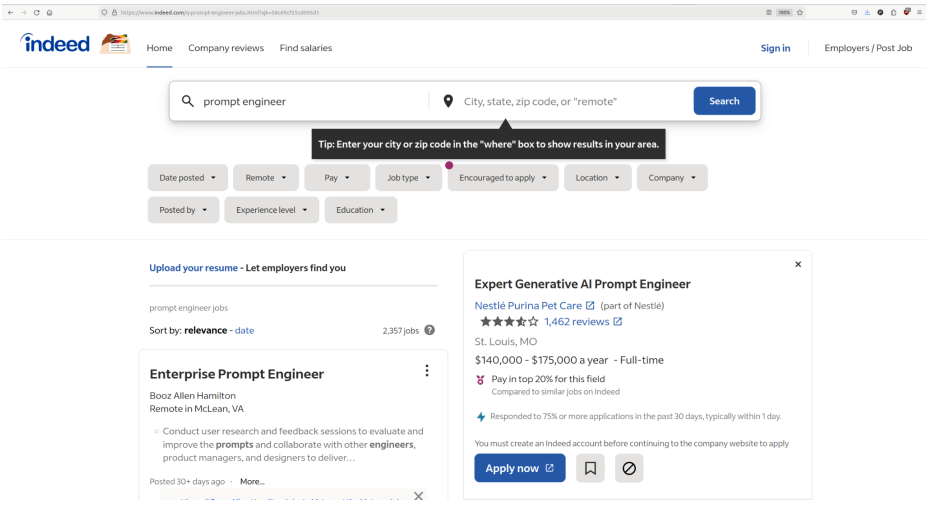

Figure 3: Search results for "Prompt engineer," Indeed [28].

3.3 RQ3: What are the benefits associated with using Generative AI in Software Engineering?

While the adoption of GenAI in the software engineering landscape is still in a rather adolescent phase, great strides have already been made in utilising the benefits that it can bring to companies in many aspects of the SDLC.

Requirements Engineering
Requirements engineering (RE) is one of the most vital and challenging stages of software engineering. This is mainly due to downfalls in communication between what the stakeholders envision the project they want to achieve and what the developers interpret those wants to be based on discussions they have with the stakeholders. RE is a time-consuming process that will more than likely need to be iterated on as the project continues over time. GenAI can play a role in mitigating the time needed to perform tasks such as elicitation, validation, and specification.

Arora, Grundy and Abdelrazek conducted a preliminary evaluation on how well ChatGPT can be integrated into the requirement elicitation stage of RE. Data was collected through four two-hour meetings with three experts working on a real-world system. These meetings outlined the project, system users, user requirements and software features. Four participants were then given the project outline, two of whom were well-established software engineers and the other two were NLP and RE research students. Over the course of the next 45 minutes, they were told to use ChatGPT to generate user-story-style requirements based on their given project outline. The results were then compared to 20 key requirements outlined by the experts of the real-world-system and given a precision and recall percentage. The precision results ranged from a high of 82% to a low of 15% and the recall results ranged from 58% to 20%. The highest recall and precision were both achieved by the same participant who was an expert, while the two participants who were students had much lower precision and recall percentages [29].

Overall, this study shows the promising benefits of GenAI in the RE landscape. These results were achieved without a model fine-tuned for this type of task but just by using ChatGPT. It also showed that experience will still play an important role in extracting the most potential benefits that GenAI can offer in assisting in RE processes such as elicitation.

Story Point Estimation
Story point estimation is an agile software development method that designates a number (points) such as 1, 2, 3, 5, etc., to user stories based on a mixture of the story's complexity and the time needed to complete the implementation of the story. This method is prone to error, even by experts in the industry. Fu and Tantithamthavorn proposed GPT2SP, a GenAI-based story point estimation technique that utilises a GPT-2 trained language model with a GPT-2 based transformer architecture. This technique was seen to be able to be used effectively across multiple projects, rather than deep-

learning methods such as Deep-SE, which are not as transferable because they are built from a project's story point dataset. Compared to 10 other baseline approaches in 16 open-source projects, GPT2SP was 34%-57% more accurate for within-project estimations and 39%-49% more accurate for cross-project estimations [30].

These findings show that this proposed model would greatly improve the accuracy of story point estimation and could be widely adopted by software engineering companies among a range of their projects rather than having to build and train a specific model for one project that is non-transferable. The proposed model could also be seen to have even more benefits if integrated with newer GPT models such as GPT-4.

Code Completion
Code completion tools such as GitHub CoPilot, Tabnine and Kite have become much more widely adopted by developers in recent years, leading to an increase in speed of development and productivity. A study conducted by GitHub showed that developers utilising CoPilot accept 30% of code suggestions, representing real productivity gains, with the percentage increasing over time as users adjust to using GitHub CoPilot. The study also showed that code generated through code completion has seen an increase in acceptance rate by reviews over six months of developers using CoPilot, with the trend steadily increasing each month [31]. An assessment done by GitHub in 2022 saw two groups of individual developers creating a HTTP server in JavaScript. One group had access to CoPilot while the other group did not. The assessment concluded that the group with CoPilot access were on average 55.8% faster at implementing the server [32].

These studies point to a correlation previously mentioned between the use of code completion tools such as CoPilot and an increase in speed of delivery and productivity in developers. This holds the promise of major economic benefit to companies as the delivery times for projects could shorten (and with fewer project resources).

3.4 RQ4: What are some of the notable risks that come with using Generative AI in Software Engineering?

As discussed previously, GenAI provides developers with a powerful tool that can speed up development, however there are risks associated with GenAI that developers should be cautious about.

Hallucination
One major problem with using GenAI in software development is that the model may hallucinate when giving a response. Hallucination in GenAI refers to "text generated by GPTs that contains factual inconsistencies, contradictions, or content that diverges from human cultural norms and expectations, despite being coherent and seemingly realistic" [33]. The exact cause of hallucination is debated, it could be that because GPT models train on entries from across the internet, the line between true facts and

misinformation is heavily blurred. Yann LeCun, a pioneer in deep learning, argues that the cause lies deep within the LLM's architecture design, saying that "there is a more fundamental flaw that leads to hallucinations" [34]. In software development, these hallucinations manifest in unreliable code generation, of which there are two main types of hallucination:

- ***Intrinsic Hallucinations:*** This is when the information is present in the source documents but its synthesis misrepresents the source and fabricates inconsistent facts that may hold true in the original source but taken out of context, it loses its true meaning [33] [35]. In software development, source code may be deemed appropriate and functional in the training set, but taken out of context can cause the code to lose its original value.

- ***Extrinsic Hallucinations:*** This is when the content can neither be validated nor contradicted, leading to assumed truths being presented as facts, this is caused by documents with poor information [33] [35]. When it comes to code generation, the model may present code that is technically accurate in the scope of its knowledge but it hallucinates inaccurate code that it interprets to be true but has nothing to support or contradict its validity.

The response of a model is highly dependent on the prompts it's given, differing prompts may render different responses, with vague prompts leading to hallucination. One proposed methodology for prompt engineering is Goal Prompt Evaluation Iteration (GPEI), depicted in Figure 4 [36]. Using this methodology addresses the issue of hallucinations by avoiding potentially vague prompts, leaning into the process of setting clear and concise goals, and of iterating on prompts to get the most optimal answer.

One mitigation strategy proposed by OpenAI is to apply Reinforcement Learning from Human Feedback (**RLHF**) to improve ethical compliance of GPT models [33]. This strategy involves developers iterating on smaller datasets by comparing the desired output by that produced by the GenAI model, labelling the output through human feedback in order to guide the model into producing a more desirable output [37].

Security

Studies have shown that using AI assistants in software development increases the likelihood of introducing security risks to the system being implemented [38]. As stated before, ChatGPT trains on data from the internet so the lack of quality control means that industry standards of cyber security may be overlooked in favour of giving a simpler and direct response that works, but one that may introduce security flaws. This may be a critical problem when dealing with high dependency systems that require code to be as close to perfect as possible. AI assistance may incur insecure code due the GPT model overlooking certain areas, not getting the full scope of the project or having a lack of knowledge surrounding the various dependencies and complexities involved in development. GPT models can help programmers detect security flaws, but it is heavily

reliant on the prompts provided by the user and also the extent of security experience the user has [38].

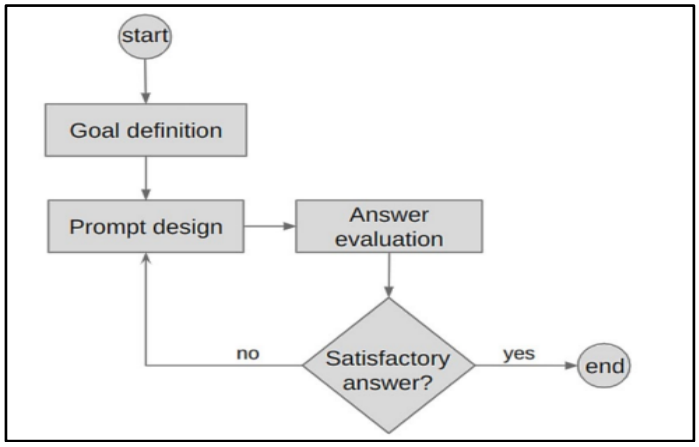

Figure 4: GPEI Methodology [36].

Plagiarism

GPT models provide quick and personalised responses, which many developers favour over the traditional method of spending time searching the web. Skipping this step may lead to overlooked plagiarism concerns [39]. Plagiarism is the act of taking an idea and passing it as your own, when developers take code from community sites like Stack Overflow, they can reference the source of the code and refer back to it when necessary. However, GPT models take, pull and mix responses from various sources causing the original code owner to be left uncredited for his work [40]. Not only is this unfair to the original code creator, but also the credibility, reliability or contextuality of the original code is to be questioned, leading once again to possible faulty code generation.

Many major ethical concerns surrounding AI are related to privacy. Many companies are concerned with how AI can infringe the private data of individuals or company documents by gathering, analysing and processing their information [41]. This poses a major risk when the code ownership is concerned. Many developers have their own coding style that is unique to them and so, their work and likeness may be used by AI assistants as training data, copying and rewrapping their work without referencing the source.

Bias

LLMs are developed and trained on vast quantities of data taken from all across the internet, some of which is imbued with inherent biases and preconceptions. Biases specifically in GenAI refers to wrongly depicted data that carries preconceptions which may influence responses and cause skews in the data used to train the model [41]. Any mistake that is found in the training data will be amplified by the GenAI model, causing it to exhibit discriminatory concerns which can significantly impact that model's

performance and generalizability [39]. This is particularly evident in code generation when mistakes are amplified and are found to carry security flaws that could negatively impact the software development process through unreliable code [39].

Cost
The cost to train and develop a LLM can pose a risk to companies if they are not prepared for the scaling factor of training models. OpenAI offers plans where you pay for the number of tokens you use. A news report states that the "process of training a large language model such as OpenAI's GPT-3 could cost more than $4 million" [42]. The same news report mentions Latitude, a small startup, was spending almost $200,000 per month on GPT-3 API calls and AWS on their GenAI-based web game, 'AI Dungeon'. Software companies are naturally aware of the cost of developing and maintaining high quality software but it is important to note the potential monetary risks that integrating AI may incur.

These risks range from trivial annoyances to potentially devastating flaws and security breaches. Faulty code is easy to miss, and for the time being, there isn't a concrete cure to AI hallucinations. Developers should be cautious using code generated by AI assistants as its user-friendly natural language processing may conceal potential flaws, contradictions and vulnerabilities found in the code.

4 Research Limitations

This research highlights GenAI's potential to improve efficiency and productivity in development processes. However, it is crucial to recognise the rapidly changing nature of this technology, which challenges the long-term relevance of current findings.

The research aim was to clearly define the broad capabilities and limitations of GenAI tools in improving software development practices rather than the specific methods software engineers use to apply GenAI in addressing more complex development issues. A clear understanding of these tools is essential, regardless of how GenAI evolves in the future. This approach ensures that the integration of GenAI into software engineering is informed by a comprehensive grasp of its potential impacts, benefits, and limitations. Despite our efforts to conduct a comprehensive and diverse study, our research was limited by factors such as time constraints, limited access to reputable material on training costs and ethical considerations of GenAI, limited empirical evidence on the real-world benefits due to how recent the adoption of GenAI is the SDLC.

It is important to also highlight that the main body of this research was conducted by a team of four final year undergraduate researchers over a fixed time period of 6 weeks. This will inevitably affect the depth of the research, the associated interpretations, and to some extent the full implementation of the MLR methodology, for example in reaching robust decisions on literature inclusions. Every effort was taken on weekly basis to

reduce the effects of these constraints through direct supervision from a senior academic.

5 Directions for Future Research

In this research we have identified the benefits of using GenAI tools in software development and the known risks associated with using it. Future research could further explore the impact that GenAI could have on the cost of building a software system, whether it is financially more cost effective or not. In the software development life cycle, it is very difficult to strike a perfect balance between cost, quality, and speed; oftentimes, one must be sacrificed over the other. We could investigate the possible ways that GenAI could improve speed and quality while reducing cost. Several studies have attempted to find this but due to it being such a novel concept there is little concrete evidence.

In addition, future research could be conducted to understand the ethical and moral concerns that arise regarding job security in software engineering. Could AI assistants potentially replace programmers altogether? While GenAI is currently highly dependent on the user, there is no telling how things will look in a few years, given the rapid advancements of AI. This warrants further research into the perspectives of other experts on the field regarding job security. Broader questions also arise: Does the adoption of GenAI have negative effects on the environment and increase inertia around progress towards greener software engineering? [43] Furthermore, what about the biases that might exist in LLM generated code [44], and how might this affect system acceptance and performance, especially for socially-critical systems? [45] These, and other important questions, are yet to be fully evaluated.

6 Conclusion

The role of a software engineer is a constantly changing one [46], and the ability of firms to successfully identify changing contexts [47] and apply appropriate adaptations [48] is reported to be important, even in smaller firms [49]. This study explores the presently evolving impact of GenAI technology on the software engineering industry, noting the reported benefits and risks associated with the technology. The findings of this study have a number of practical implications. Specific areas were identified within the software development life cycle where GenAI tools could have the significance, including Specification Generation, Just-In-Time Developer Feedback, Improved Testing, Documentation, and Language Translation. The findings furthermore suggest that the usage of GenAI tools applied in a software engineering context can have many benefits, such as from higher developer fulfilment, increases in story-point estimation accuracy, and major increases in code implementation speeds.

While the adoption of GenAI in software engineering is receiving considerable attention, it is important to note that there are a variety of concerns that should not be

casually overlooked. Although GenAI can ostensibly increase productivity, there are issues concerning the accuracy of the data returned from GenAI platforms. There are furthermore a range on not insignificant risks associated with security, plagiarism and bias. This research also highlights that there are specific monetary and safety implications to be addressed if adopting GenAI.

GenAI represents a significant technological step forward, but its adoption in software engineering is evolving and it is not risk free. In a world of hybrid working, the ability of firms to softly monitor employee activity is significantly reduced [50], with the result that companies may inadvertently incorporate GenAI code into their products where developers adopt the technology while working remotely (and without informing the company). Valuable intellectual property might also be put at risk where it is submitted into GenAI platforms, for example where source code is submitted to get a summary of the functionality. The challenge for firms is to balance GenAI's adoption while also continuing to address other key business considerations including, intellectual property management, quality, and predictability. GenAI adoption may also throw a considerable focus back on requirement engineering, which in GenAI terminology equates to Prompt Engineering. Navigating the adoption of GenAI based software engineering will be a significant and evolving challenge for firms in the coming years.

Acknowledgements. This research is supported in part by SFI, Science Foundation Ireland (https://www.sfi.ie/) grant No SFI 13/RC/2094_P2 to Lero - the Science Foundation Ireland Research Centre for Software.

References

[1] V. Brühl, "Generative Artificial Intelligence – Foundations, Use Cases and Economic Potential," Intereconomics, vol. 59, no. 1. Walter de Gruyter GmbH, pp. 5–9, Feb. 01, 2024. doi: 10.2478/ie-2024-0003.

[2] S. Kumar, D. Musharaf, S. Musharaf, and A. K. Sagar, "A Comprehensive Review of the Latest Advancements in Large Generative AI Models," Communications in Computer and Information Science. Springer Nature Switzerland, pp. 90–103, 2023. doi: 10.1007/978-3-031-45121-8_9.

[3] Feuerriegel, S., Hartmann, J., Janiesch, C. et al. Generative AI. Bus Inf Syst Eng 66, 111–126 (2024). https://doi.org/10.1007/s12599-023-00834-7

[4]Tomczak, J. M. (2022). *Deep generative modeling*. Springer International Publishing. https://doi.org/10.1007/978-3-030-93158-2

[5] S. Alfasly, P. Nejat, S. Hemati et al., "When is a Foundation Model a Foundation Model," https://arxiv.org/pdf/2309.11510.pdf

[6] Jung KH. Uncover This Tech Term: Foundation Model. Korean J Radiol. 2023 Oct;24(10):1038-1041. doi: 10.3348/kjr.2023.0790. PMID: 37793672; PMCID: PMC10550749.

[7] C. Moore, T. Taylor, C. Anderson, "Exploring the Frontiers of Generative AI: From ChatGPT to Multimodal Innovations," https://www.researchgate.net/profile/Charles-Anderson-32/publication/376831367_Exploring_the_Frontiers_of_Generative_AI_From_ChatGPT_to_Multimodal_Innovations/links/658b469a3c472d2e8e90733a/Exploring-the-Frontiers-of-Generative-AI-From-ChatGPT-to-Multimodal-Innovations.pdf

[8] Banh, L., Strobel, G. Generative artificial intelligence. Electron Markets 33, 63 (2023). https://doi.org/10.1007/s12525-023-00680-1

[9] Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. Electron Markets 31, 685–695 (2021). https://doi.org/10.1007/s12525-021-00475-2

[10] Sagar, S., Hongyi, Z., Balaji, P., Yidong, C., Hsinchun, C. & Jay F. Nunamaker Jr. (2023) Deep Learning for Information Systems Research, Journal of Management Information Systems, 40:1, 271-301, DOI: 10.1080/07421222.2023.2172772

[11] Y. Cao, S. Li, Y. Liu, "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT," https://arxiv.org/pdf/2303.04226.pdf

[12] H. Naveed, A. Ullah Khan, S. Qiu, "A Comprehensive Overview of Large Language Models," https://arxiv.org/pdf/2307.06435.pdf

[13] K. S. Kalyan, A. Rajasekharan, S. Sangeetha, "AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing," https://arxiv.org/pdf/2108.05542.pdf

[14] L. Ouyang, J. Wu, X. Jiang et al. "Training language models to follow instructions with human feedback," https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf

[15] V. Sorin, E. Klang, "Large language models and the emergence phenomena", European Journal of Radiology Open, Volume 10, 2023, 100494, ISSN 2352-0477, https://doi.org/10.1016/j.ejro.2023.100494.

[16] Chui, Michael, Eric Hazan, Roger Roberts, Alex Singla, Kate Smaje, Alex Sukharevsky, Lareina Yee, and Rodney Zemmel. "The Economic Potential of Generative AI: The Next Productivity Frontier." McKinsey & Company, June 14, 2023. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier

[17] Generative AI: From Buzz to Business Value. Accessed February 23, 2024. https://kpmg.com/kpmg-us/content/dam/kpmg/pdf/2023/generative-ai-survey.pdf

[18] Pothukuchi, Ameya Shastri, Lakshmi Vasuda Kota, and Vinay Mallikarjunaradhya. "Impact of Generative AI on the Software Development Lifecycle (SDLC)." SSRN, August 22, 2023. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4536700

[19] Dohmke, T.. "The Economic Impact of the AI-Powered Developer Lifecycle and Lessons from Github Copilot." The GitHub Blog, June 27, 2023. https://github.blog/2023-06-27-the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot/

[20] I. Ozkaya, "Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications," in *IEEE Software*, vol. 40, no. 3, pp. 4-8, May-June 2023, doi: 10.1109/MS.2023.3248401

[21] John E. Robert and Douglas Schmidt (Vanderbilt University). "Generative AI Q&A: Applications in Software Engineering." SEI Blog, November 16, 2023. https://insights.sei.cmu.edu/blog/generative-ai-question-and-answer-applications-in-software-engineering/

[22] M. Islam, F. Khan, S. Alam and M. Hasan, "Artificial Intelligence in Software Testing: A Systematic Review," *TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)*, Chiang Mai, Thailand, 2023, pp. 524-529, doi: 10.1109/TENCON58879.2023.10322349.

[23] L. Layman and R. Vetter, "Generative Artificial Intelligence and the Future of Software Testing," in *Computer*, vol. 57, no. 1, pp. 27-32, Jan. 2024, doi: 10.1109/MC.2023.3306998.

[24] C. Ebert and P. Louridas, "Generative AI for Software Practitioners," in *IEEE Software*, vol. 40, no. 4, pp. 30-38, July-Aug. 2023, doi: 10.1109/MS.2023.3265877.

[25] OpenAI, “Introducing Gpts.” November 6, 2023, Accessed: February 23, 2024. https://openai.com/blog/introducing-gpts

[26] AWS, “What is prompt engineering? - ai prompt engineering explained”, Accessed February 23, 2024. https://aws.amazon.com/what-is/prompt-engineering/

[27] D. C. Schmidt, J. Spencer-Smith, Q. Fu, and J. White, “Towards a Catalog of Prompt Patterns to Enhance the Discipline of Prompt Engineering,“ https://www.dre.vanderbilt.edu/~schmidt/PDF/ADA-User-Journal.pdf

[28] "Search Results for 'Prompt Engineer' Jobs," Indeed. Accessed: Feb. 22, 2024. Available: https://www.indeed.com/q-prompt-engineer-jobs.html

[29] C. Arora, J. Grundy and M. Abdelrazek, “Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs”, https://arxiv.org/pdf/2310.13976.pdf

[30] M. Fu and C. Tantithamthavorn, "GPT2SP: A Transformer-Based Agile Story Point Estimation Approach," in *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 611-625, 1 Feb. 2023, doi: 10.1109/TSE.2022.3158252

[31] T. Dohmke, “Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle” https://arxiv.org/ftp/arxiv/papers/2306/2306.15033.pdf

[32] S. Peng, E. Kalliamvakou, P. Cihon, M. Demirer, “The Impact of AI on Developer Productivity: Evidence from GitHub Copilot,” https://arxiv.org/pdf/2302.06590.pdf

[33] A Culturally Sensitive Test to Evaluate Nuanced GPT Hallucination | IEEE Journals & Magazine | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10319443 (accessed Feb. 23, 2024).

[34] C. Smith, “ChatGPT’s Hallucinations Could Keep It from Succeeding - IEEE Spectrum,” spectrum.ieee.org, Mar. 13, 2023. https://spectrum.ieee.org/ai-hallucination

[35] K. Ando, T. Okumura, M. Komachi, H. Horiguchi, and Y. Matsumoto, “Is artificial intelligence capable of generating hospital discharge summaries from inpatient records?,” *PLOS Digital Health*, vol. 1, no. 12, p. e0000158, Dec. 2022, doi: https://doi.org/10.1371/journal.pdig.0000158.

[36] J. D. Velásquez-Henao, C.J. Franco-Cardona & L. Cadavid-Higuita, “Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering”, https://revistas.unal.edu.co/index.php/dyna/article/view/111700/90275

[37] P. P. Ray, “ChatGPT: a Comprehensive Review on background, applications, Key challenges, bias, ethics, Limitations and Future Scope,” *Internet of Things and Cyber-Physical Systems*, vol. 3, no. 1, pp. 121–154, Apr. 2023, doi: https://doi.org/10.1016/j.iotcps.2023.04.003.

[38] N. Perry, M. Srivastava, D. Kumar, and D. Boneh, “Do Users Write More Insecure Code with AI Assistants?,” *arXiv (Cornell University)*, Nov. 2022, doi: https://doi.org/10.1145/3576915.3623157.

[39] “Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications | IEEE Journals & Magazine | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/abstract/document/10109345 (accessed Feb. 23, 2024).

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10109345

[40] J. Dien, “Editorial: Generative Artificial Intelligence as a Plagiarism Problem,” pp. 108621–108621, Jun. 2023, doi: https://doi.org/10.1016/j.biopsycho.2023.108621.

[41] V. D. Kirova, C. S. Ku, J. R. Laracy, and T. J. Marlowe, “The Ethics of Artificial Intelligence in the Era of Generative AI,” *Journal of Systemics, Cybernetics and Informatics*, vol. 21, no. 4, pp. 42–50, Dec. 2023, doi: https://doi.org/10.54808/jsci.21.04.42.

[42] J. Vanian and K. Leswing, “ChatGPT and Generative AI are booming, but at a very expensive price,” *CNBC*, Mar. 13, 2023. https://www.cnbc.com/2023/03/13/chatgpt-and-generative-ai-are-booming-but-at-a-very-expensive-price.html

[43] Freed, M., Bielinska, S., Buckley, C., Coptu, A., Yilmaz, M., Messnarz, R., Clarke, P.M.: An Investigation of Green Software Engineering. In Proceedings of Systems, Software and Services Process Improvement, EuroSPI 2023, 29-31 August 2022, Grenoble, France. Communications in Computer and Information Science (CCIS), vol 1881. Springer, Cham.

[44] De Buitlear, C., Byrne, A., McEvoy, E., Camara, A., Yilmaz, M., McCarren, A., Clarke, P.M.: Investigating Sources and Effects of Bias in AI-Based Systems – Results from an MLR. In Proceedings of Systems, Software and Services Process Improvement, EuroSPI 2023, 29-31 August 2022, Grenoble, France. Communications in Computer and Information Science (CCIS), vol 1881. Springer, Cham.

[45] Dagg, N., Kostick, C., Fallon, J., O'Neill, A., Yilmaz, M., Messnarz, R., Clarke, P.M.: Socially-Critical Software Systems: Is Extended Regulation Required? In: Proceedings of Systems, Software and Services Process Improvement, EuroSPI 2022, 30 August 2022 - 2 September 2022, Salzburg, Austria. Communications in Computer and Information Science (CCIS), vol 1646. Springer, Cham. pp. 610–622. Available: https://doi.org/10.1007/978-3-031-15559-8_43

[46] Meade E., O'Keefe, E., Lyons, N., Lynch, D., Yilmaz, M., Gulec, U., O'Connor, R.V., Clarke, P.M.: The Changing Role of the Software Engineer. In: Proceedings of the 26th European and Asian Conference on Systems, Software and Services Process Improvement (EuroSPI 2019), Springer CCIS Vol. 1060, pp.682-694, 18-20 September 2019, Edinburgh, Scotland.

[47] Clarke, P., O'Connor, R.V.: Changing situational contexts present a constant challenge to software developers. In: Proceedings of the 22nd European and Asian Conference on Systems, Software and Services Process Improvement (EuroSPI 2015), CCIS (Vol. 543), pp. 100-111, 30 September - 02 October 2015, Ankara, Turkey. (2015)

[48] Clarke, P., O'Connor, R.V.: An Approach to Evaluating Software Process Adaptation, In: Proceedings of the 11th International Conference on Software Process Improvement and Capability dEtermination (SPICE 2011), CCIS Vol. 155, pp.28-41. Heidelberg, Germany: Springer-Verlag. May 2011.

[49] Marks, G., O'Connor, R.V., Clarke, P.: The impact of situational context on the software development process - A case study of a highly innovative start-up organization. In: Proceedings of the 17th International SPICE Conference (SPICE 2017), pp.455-466; 4-5 October 2017, Palma de Mallorca, Spain.

[50] Clarke, P.: "The Remote Working Genie Is Out of the Office Bottle", in IEEE Software, vol. 40, no. 4, pp. 88-95, July-Aug. 2023, doi: 10.1109/MS.2023.3258921