2023

IS AI ABOUT TO REVOLUTIONISE SOFTWARE DEVELOPMENT?

By Matthew Belcher
and Danish Javed

codurance

A Lifting Tide for All

In the rapidly evolving world of technology, it seems we are entering a new era with the advent of Generative Artificial Intelligence (or Gen AI as it is more commonly known). Gen AI promises to redefine the way we approach software development, automating mundane coding tasks, generating complex code structures and improving software quality. Given this, is Gen AI now poised to revolutionise software development? Is that just hype or are we truly on the cusp of something transformational?

Gen AI tooling could drastically increase productivity, reduce human error, and unleash a wave of innovation by providing developers the freedom to focus on high-level strategic thinking and creative problem-solving. This revolutionary approach invites us to reimagine software development not just as a human-driven discipline, but perhaps a symbiotic relationship between human intellect and artificial intelligence, where the whole far surpasses the sum of its parts. In this article, we'll take a closer look at some of these Gen AI tools to understand what impact they might have on software development and how they might lead us into some different ways of working. We'll reflect on some of the broader legal and ethical implications as well as considering what it might mean for Software Craftsmanship.

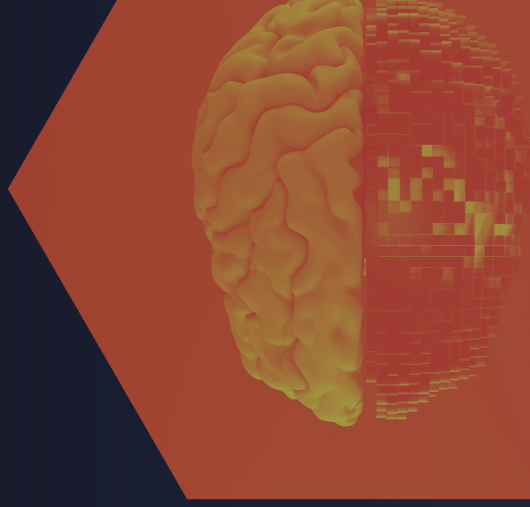
What is Generative AI

Generative AI is a type of artificial intelligence technology that can produce various types of content, including text, imagery, audio, video. All of this is possible from having consumed huge amounts of training data from various sources, spotting patterns and being able to generate new data based on these observed patterns, and synthetic data from a given input and in particular in response to prompts.

This art of giving precise prompts is a sub field of data science and machine learning also known as Prompt Engineering. Prompt engineering is about giving clear instructions for getting the best help. It makes the assistance smarter and more effective for various tasks. The more concise the prompts, the better the outcome from the Gen AI tools.

Generative AI models learn the patterns and structure of their input training data and then generate new data that has similar characteristics. Generative AI has potential applications across a wide range of industries, including art, writing, software development, healthcare, finance, gaming, marketing, fashion, administration and regulation within the public sector.

One thing to bear in mind is that Gen AI models are not general purpose models applicable to everything - moreover it does not mean General Intelligence. They lack common sense, emotional intelligence and are solely designed to generate new data that resembles a given dataset.



Chat GPT

OpenAI's generative pre-trained transformer or, ChatGPT as it is more commonly known, is not strictly touted as a software development tool. It is more a general-purpose user interface on top of a machine learning language model that is able to generate human-like textual output in response to inputs it receives. Infact, the model is so vast it is referred to as a Large Language Model (or LLM).

However, despite not being aimed at software development specifically, it has certainly had an impact on how software developers approach writing software. It provides a software developer with an interface to "talk" through a coding problem. ChatGPT is then able to respond, suggesting prompts and code snippets for the engineer to use. It is important to keep in mind, the responses ChatGPT provides are generated using the vast amount of data it has collected and been trained on from the Internet, primarily.

The most recent iteration of the underlying model GPT4.5 (at the time of writing) has brought various improvements to the responses it can give. These can then be used for a range of tasks related to software development such as improving documentation, generating test cases, critiquing code and of course generating code snippets.

ChatGPT can be used for a range of different use cases within software development. It can provide assistance in understanding a programming language or framework, allowing developers to quickly access relevant information and code snippets. It generates code and offers suggestions, speeding up prototyping and saving time on routine tasks. ChatGPT can assist in error detection and debugging, providing insights and potential fixes. It also facilitates rapid prototyping and exploration, enabling developers to experiment with different ideas. Additionally, its natural language interface enhances accessibility for users with varying levels of programming expertise.

At Codurance, we champion Extreme Programming (XP) practices in the software development work with our clients. One of these practices is Pair Programming. However, we take a pragmatic approach to this - not everything requires Pair Programming. With the advent of ChatGPT and similar tools, it is possible to mimic some of the benefits of Pair Programming.

While ChatGPT clearly has a lot to offer within software development, there are also limitations to consider. It may lack contextual understanding depending on the plugin being used, leading to irrelevant or inaccurate suggestions that require human validation. The model's dependency on training data can introduce biases or flawed code examples. It may not fully comprehend the developer's intent or project context, resulting in partially addressing desired functionality.

Privacy and security concerns may arise if sensitive information is inadvertently shared. Ethical considerations, such as avoiding biassed or discriminatory code, need attention. Over reliance on ChatGPT may hinder skill development, and real-time collaboration is still essential for nuanced discussions.

GitHub Copilot

GitHub Copilot is the latest in a series of tools making waves in the software development industry. We were pleasantly surprised by its speed of suggestions and correctness. Our experience of using this together with the Test Driven Development (TDD) approach worked well. It was almost like having another person pair with us. The generated code snippets were not always perfect and often needed tweaking but they got us part of the way there to avoid the mundane setup.

When a test-case is named well, Copilot will aim to fill the body of the test based on our method naming and utilising the code context. While this is not the typical usage scenario of Copilot exhibited from various coding demonstrations i.e. providing step-by-step instructions or prompts it did well to create the body of the test.

The latency of suggestions was quite good, it was noticeable at times but generally was a good experience. We didn't think that cycling through suggested snippets was as natural as other options.

Copilot is also under heavy development with a lot of features being released on a regular basis. One of the newer additions to the plugin is the chat feature (Only available in VSCode at the time of writing) as well as the Copilot Labs that have the concept of Brushes e.g. Refactor, Cleanup, Readable etc. that aim to make your code cleaner, readable or help to refactor to an alternative.

Another interesting feature in Copilot Labs is that you can translate code from one language to another e.g. from Java to Python at a snippet level. This indicates a possibility in the future to translate an entire application from one language to another, which would prove to be useful if someone wants to port their codebase to an alternative language. Perhaps as part of a wider journey to evolve their technical stack. This could be especially useful for organisations wishing to undertake a software modernisation initiative involving older codebases or languages.

AWS Code Whisperer

Amazon AWS has been working on their own software development AI assistant. Similar to Copilot it supports many languages, popular IDEs and addresses the same needs of AI assisted code generation. Just like our TDD approach with Copilot, we were able to get method completion from a well named testcase.

The implementation details of CodeWhisperer are not well known but it's been trained on code that's open-source or internal to Amazon. It's currently being offered for free for individuals with an AWS Builders account and licensed version as CodeWhisperer Professional for organisations currently making use of AWS Cloud services.

It is important to note that CodeWhisperer is also the same as Copilot where it can help write applications but, an area where CodeWhisperer really shines is when writing code for AWS services. Some of these commonly used services include AWS S3, Lambda etc. Copilot can do the same but CodeWhisperer does a better job of doing so.

Another out of the box feature is that Code Whisperer will also check for security vulnerabilities listed in the OWASP Top 10 against the code being generated or written by the engineer. Once a security vulnerability is identified, CodeWhisperer will provide suggestions on how to fix them. However, this functionality is limited to Python, Java & JavaScript at this time.

Overall, based on our experience to-date, we believe that CodeWhisperer in its current state is not as feature-rich as Copilot but this could of course change in the future as this is a fast moving space, we continue to monitor both tools actively.

Privacy & Security Concerns

Privacy and security are crucial aspects to consider when using tools like GitHub Copilot and CodeWhisperer. While these AI-driven coding assistants aim to provide efficient and accurate suggestions, it is important to note that there is a possibility of generating code with security vulnerabilities, despite efforts to filter out insecure code from their training data.

It is essential to acknowledge that the emergence of these AI tools presents security implications that need to be addressed. As these tools become more sophisticated, it becomes imperative to have robust security measures in place to safeguard sensitive information and prevent potential misuse.

An important thing to consider is that as users of these tools, we retain ownership of the code generated. This also includes any suggestions offered by these tools, as these are customised to our specific requirements based on the prompts we provide.

GitHub Copilot and CodeWhisperer both use data collection to enhance their services. GitHub Copilot captures real-time prompts, AI-generated suggestions, and user engagement data within the IDE. However, GitHub Copilot for Business doesn't store prompts and suggestions, only pseudonymous user engagement data.

Similarly, CodeWhisperer (individual) stores usage and content data to improve its service, but users can opt out if they prefer not to share this data. The collected data includes client-side telemetry and content, but at the professional tier, content collection for service improvement is disabled. Rest assured, telemetry doesn't include actual code or personally identifiable information (PII), and data usage can be limited to your own VPC for better control.

Although both tools lack access to the source code, they may still have access to contextual information like the tech stack being used and domain feature descriptions provided through prompts. To avoid any intellectual property issues, consulting with legal advisors is recommended. Understanding these privacy and security considerations helps us to make informed decisions and to utilise these powerful tools while safeguarding our code and personal information.

Software Craftsmanship & AI

Those that are familiar with Codurance will know that the principles of Software Craftsmanship are at the very core of the work we do. So it would be remiss of us to not consider this when discussing how AI might impact Software Craftsmanship. For those unfamiliar with the principles of Software Craftsmanship, they are designed to support the overall goal of raising the bar of professional software development within the industry.

It is our belief that even with the rise of AI and Generative AI code assistance tooling, Software Craftsmanship still very much has a place within software development. Whilst Gen AI code assistance tools such as the ones discussed in this article can generate code that may well be functionally correct it is important to remember that they are still just tools. As such they lack the ability to understand the nuances of the business domain, the specific needs of a user, or a deeper understanding of the wider context in which the software being written resides.

Software Craftsmanship is not just about creating functionally correct software. At the heart of it is the notion of taking pride in the software being developed and producing well-designed, maintainable and reliable software solutions. Something that a human, a Software Craftsperson must own and not outsource to a tool.

However, we do believe that Software Craftsmanship and AI can happily coexist and compliment each other. For example, Gen AI code assistance tools and other related tooling can be used to automate repetitive tasks that are well understood, not overly complex but time consuming all the same. This frees up time for software developers to focus more of their time and energy on more value-adding activities such as overall architectural design, solving complex problems and implementing business critical features.

Conclusion

At the beginning of this article, we posed the question of whether or not AI is poised to revolutionise software development. Whilst the release of Gen AI tools such as the ones discussed here, undoubtedly have the potential to increase productivity amongst other things there is still a way to go before we can declare that software development has been revolutionised.

Those that are following AI will know that this is an incredibly fast-moving space, with new tooling seemingly being released every week. This means that it is also a constantly evolving space - AI models are being trained and new models released. Each model is seemingly more advanced than the last. At the time of writing, Meta has just released of their second-generation open-source large language model Llama2. All of this means that whilst AI isn't revolutionising software development at this moment in time, it certainly does have the potential to be transformative.

What is clear is that AI is not going anywhere and as such it is vital that technology leaders start to assess how AI might impact their software development process. Those that do so will find themselves best placed to benefit from it. A part of this assessment needs to include some of the security and legal implications discussed earlier in this article as well as how current processes and technical practices may evolve or new practices be introduced to maximise the benefit of AI adoption. For example, a common theme of discussion recently has been around Test Driven Development and how AI may play a role in that practice.

It is important to understand the full implications of adopting AI tooling in your software development process. Whilst these AI tools can offer tremendous assistance, the suggestions and prompts generated by these tools are owned by the respective companies. This raises concerns about ownership and intellectual property. Software developers and technical leaders need to be mindful of this and take necessary precautions to protect their own ideas and contributions. That being said, Meta's open sourcing of their latest AI model is certainly an interesting development in this space.

As well as understanding the legal and ethical implications it is also important to realise that increased code-quantity does not necessitate quality, in fact it could be argued that we are going to see lots of ill-fitting code where early AI models have been overly relied upon. Software developers need to be practising caution when using Gen AI code assistance tooling and applying the same level of attention and care to the generated code as they do to the code they write.

At Codurance, we are cautiously optimistic by the potential of AI within software development. We are excited by the possibilities AI could open up and the productivity improvements it could bring in how software is developed going forwards. But at the same time, we remain mindful of security, legal and indeed ethical implications.

About the Authors

Matt Belcher
Principal Craftsperson

Matt has over 15 years of experience working within software development. The vast majority of that time he has spent as a consultant. Matt has worked with a large number of clients across many countries, tech stacks, languages and technical architectures. He has played numerous roles during his career from Software Developer to Solutions Architect. Nowadays Matt works alongside technical leaders from client organisations advising and supporting in areas such as technical architecture, software development and Agile practices.

Matt is passionate about helping organisations improve their software delivery capabilities, whether it be through embedding good technical practices or establishing a clear technical vision. He has a particular interest in Event Driven Architectures and Serverless.

Danish Javed
Software Craftsperson

As a seasoned Backend Developer and co-organiser of Software Crafters North, Danish is passionate about delivering good quality software solutions. Danish has honed his skills in building and maintaining high-performance web applications that are both robust and scalable. He is a firm believer in the power of TDD, clean architecture, and clean code principles to create the best possible outcomes for clients.

When it comes to working, Danish thrives in fast-paced, high-pressure environments and is committed to delivering quality results. He is also an active member of the software development community and love sharing his knowledge and experiences to help others grow. Whether it's co-organising events like Software Crafters Manchester or staying up-to-date on the latest industry developments, Danish is always looking for ways to better himself and those around him.

@codurance **hello@codurance.com** **www.codurance.com**