publicis
sapient

The Executive Guide to AI-Assisted Software Development

Why software may finally eat the world

Imagine saving 4,500 years of developer work in just one year—an achievement so astonishing it sounds improbable. Yet, in a LinkedIn article from Amazon's CEO, that's exactly what the company claims to have achieved by integrating AI into their code modernization efforts. This isn't a distant dream; it's a strong example of AI's impact on software production.

Currently, American enterprises are pouring nearly half of their IT budgets into application development and support, as highlighted in a recent study on IT spending intentions. These applications are more than just operational efficiency enablers—they're often the main way businesses connect with customers and drive differentiation. The ability to consistently deliver the right digital products, quickly and with high quality, is a significant competitive weapon. Despite Marc Andreessen's 2011 prediction that "software is eating the world," many enterprises are still grappling with the gap between their digital transformation ambitions and the harsh realities of a compressed macroeconomic environment. This tension has left executives in a difficult position.

AI-enabled software development could finally bridge this gap, offering a pathway to both reduce decades of accumulated technical debt and spark a wave of digital innovation—all without the need for increased budgets. Importantly, software development doesn't just encompass coding and technical tasks, it covers product engineering and digital business transformation as a whole. As enterprise leaders explore the potential of artificial intelligence (AI) in the software development lifecycle (SDLC), including the full scope of activities involved in crafting digital solutions that deliver tangible value, they're faced with a landscape rich in opportunity but also rife with complex questions:

- **How can I use AI to supercharge the way digital products are made?**

- **How do I safeguard intellectual property and ensure security in this new era?**

- **Which tools, methods and talent should I prioritize as things continue to evolve?**

- **How do I measure progress?**

At Publicis Sapient, we lead large-scale digital transformations for our clients, drawing on our deep, hands-on experience. We've engaged in dozens of enterprise AI deployments to speed application delivery and modernization. We've invested $325 million to transform our parent company, Publicis Groupe, with AI-enabled software we call "CoreAI." We are using our CoreAI to rigorously test, learn and distinguish what truly works in our AI-enabled SDLC from theoretical possibilities. These efforts have equipped us with unique insights and practical strategies to help enterprises navigate the AI revolution in software development.

In this article, we'll explore the transformative potential of AI in software development, share our most valuable lessons and answer the critical questions that business and technology executives are asking today.

Welcome to the future of software development—are you ready to lead?

TABLE OF CONTENTS

01 Understanding AI-assisted software development

The concept of AI generating code isn't new. Since Microsoft launched GitHub Copilot in November 2021, developers have been receiving real-time code suggestions and assistance, seamlessly integrated into their workspace tools. With over 1.3 million paid subscribers and 50,000 enterprise customers, GitHub Copilot has quickly become the most widely adopted AI-powered developer assistant.

But AI-assisted software development is more than just a code suggestion tool in the developer's toolkit or a conversational chatbot like ChatGPT. Its potential is far more profound—by eliminating complexity and human toil, enterprises can slash technical debt and accelerate transformation.

Claude 3.5 Sonnet and OpenAI o1—leading the way in coding

Consider the recent release of Anthropic's latest large language model (LLM), Claude 3.5 Sonnet, on June 20, 2024. Claude 3.5 Sonnet made headlines for its industry-leading performance across a range of benchmarks, particularly in its ability to write software code. One standout evaluation involved the model's ability to understand an open-source codebase and implement a pull request for a bug fix or new feature based solely on a natural language description.

The model was then tested on whether all the codebase tests—kept hidden from the model—would pass for the completed submission. This wasn't just a trivial exercise; the problems were based on real pull requests, requiring the model to edit multiple files, write, run and iteratively correct code in a secure, sandboxed environment without internet access. Achieving a new high-water score on this benchmark is a remarkable feat. It demonstrates AI's growing ability to autonomously manage complex, multistep software development tasks, from understanding requirements to coding and testing, all without human guidance.

The buzz surrounding Claude's release was immediate. Social media lit up with excited posts from developers and enthusiasts alike. A technical founder on Reddit claimed a 10x productivity boost using the model compared to pre-LLM days.

Another individual shared their experience in a post on X of deploying a web app using Claude, Replit and Google Firebase after just six days of learning to code.

Or, consider the more recent release of OpenAI's "o1-preview" enhanced reasoning system, formerly known as "Project Strawberry/Q," on September 12, 2024. This new model approaches problem-solving with a more deliberate, reflective process. Early tests show that o1-preview can analyze, backtrack and weigh different options before arriving at a solution, significantly reducing the likelihood of errors or inaccurate responses. This makes it well suited for complex tasks that require planning and iteration—such as enterprise-level code development or large-scale program planning. OpenAI reports that o1 has dramatically improved its Codeforces Elo benchmark performance, jumping from the 11th percentile, "Newbie," to the 89th percentile, or "Expert" level, compared to human developers.

These stories are thrilling, and it's easy to get swept up in the idea of a future where the SDLC is fully automated by AI. However, as exciting as these developments are, in an enterprise context it's crucial to separate the reality from the hype. **Understanding where AI is genuinely transforming the SDLC—and where it falls short—requires a balanced perspective.**

Definition and explanation of AI-assisted software development

AI-assisted software development refers to the integration of AI technologies, particularly LLMs, into the software development process to enhance and accelerate various aspects of business and systems analysis, design, coding, testing, deployment and maintenance. This approach leverages AI to augment labor-intensive tasks, such as analyzing existing artifacts, ideating design alternatives, identifying potential bugs, optimizing performance and generating code, test cases and documentation based on natural language descriptions and existing work products.

Differentiation between AI-assisted and traditional development methods

Traditional software engineering relies heavily on human developers. It's true that any modern SDLC already uses automation extensively: DevOps tooling automates repetitive tasks like code integration, regression testing and deployment, while static code analyzers use predetermined rulesets to analyze and grade code for compliance. Yet, even with these tools, traditional development demands deep domain and technical expertise and involves a significant amount of manual effort throughout the lifecycle.
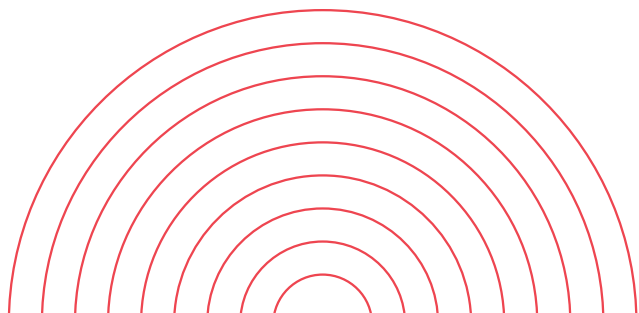
AI-assisted software development, on the other hand, integrates AI, especially LLMs, to more efficiently and accurately perform or assist with tasks that were formerly resistant to automation. LLMs are particularly powerful tools for software tasks because they have been pretrained on a vast corpus of preexisting software engineering data. Leveraging their trained models, AI tools like GitHub Copilot can suggest lines of code or even entire functions as developers type, acting as an AI-powered pair programmer. More advanced AI systems, such as Claude Sonnet 3.5 and Cognition's Devin, go a step further by understanding complex codebases and making sophisticated changes. These systems can implement new features or fix bugs based on high-level instructions by iteratively writing, modifying and testing the code.

The main difference: Traditional DevOps and other SDLC tools, like code analyzers, are rule-based and deterministic, meaning they always produce the same outputs for a given set of inputs. AI tools are probabilistic, with outputs that can vary depending on the data they were trained on and the specific inputs, such as the prompts and context they receive.

The implication: While AI-generated code can speed up development, it may not always meet the cleanliness or standards expected in an enterprise environment due to its training or the quality of inputs.

What does this mean for IT leaders? A skilled developer with a deep understanding of the business domain and development frameworks can use AI as a true force multiplier in improving efficiency and quality. Without this expert human steering the AI and critically examining its outputs, the gains are quickly lost, and either productivity or quality, or both, are compromised.

Therefore, integrating AI into the SDLC must be done with careful consideration and continuous skilled human oversight.

The **top AI use cases** for developers

According to a recent Stack Overflow survey, **the top three AI use cases** for developers are:

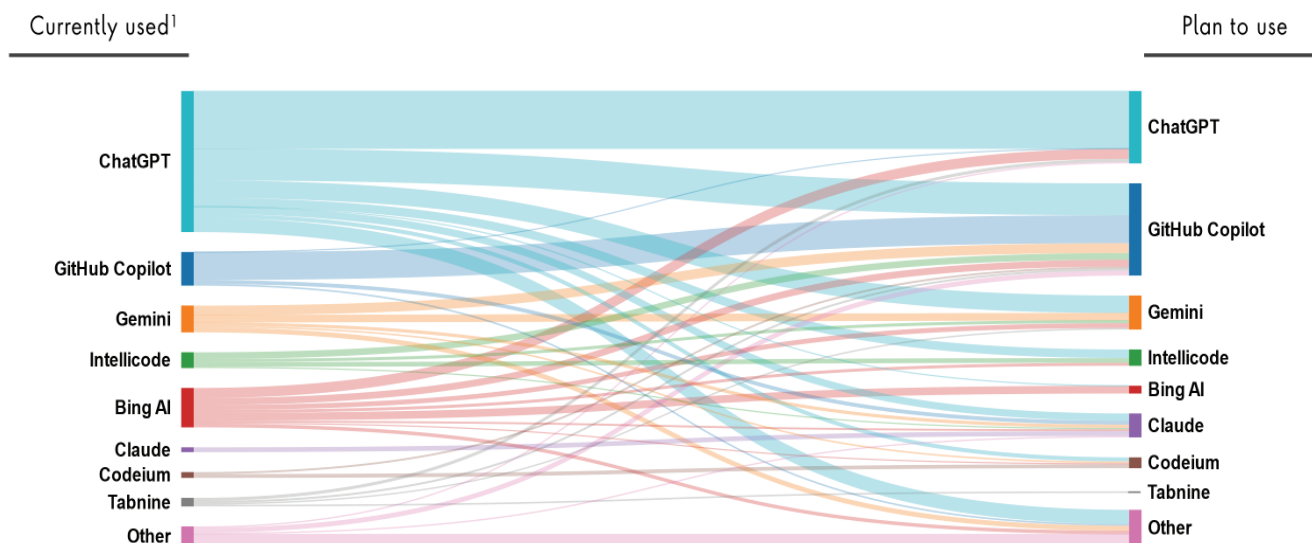- *Writing code*

- *Searching for answers*

- *Debugging*

In fact, while AI tools are often used by software developers for quick answers and assistance with small code sections, the same survey also reported that 38 percent of developers report code assistants provide inaccurate information half of the time or more. There are new code assistant tools being released every day, though ChatGPT and GitHub Copilot dominate the space, with the survey showing GitHub Copilot and Anthropic's Claude rapidly growing in usage.

While majority of developers use LLMs like ChatGPT today, GitHub seems to be preferred long-term - only Codeium and Claude maintain and grow their user-base

AI search and developers' tools used this year and preferred to use next year (# of respondents using AI, N= 45,636)

Currently used[1]

Plan to use

ChatGPT
GitHub Copilot
Gemini
Intellicode
Bing AI
Claude
Codeium
Tabnine
Other

ChatGPT
GitHub Copilot
Gemini
Intellicode
Bing AI
Claude
Codeium
Tabnine
Other

76

Note: (1) Multiple selection possible resulting on potential double counting on currently used tools, only flow with >300 answers shown
Source: https://survey.stackoverflow.co/2024/ai#developer-tools-ai-tool (Jul. 2024)

However, as we will discuss next, the potential for AI-assisted software development is much larger than developer assistants and code completion.

02 Capturing value from AI-assisted software development

The key benefits of AI-assisted software development

Many organizations may already be using AI tools for code suggestions, defect detection and bug fixing. However, the opportunity space for value creation is much larger. Currently, software development is a labor-intensive process fraught with human toil. Consider the following challenges:

Cross-disciplinary collaboration: Multiple disciplines—including Strategy, Product, Experience, Engineering and Data & AI—must work together to solve complex customer and business problems, define requirements and build systems that deliver value. This collaboration is expected to be agile and iterative, yet the complexity, scale and handoffs often hinder progress.

Cognitive overload: Architects, product and engineering leaders and developers are burdened with an unreasonable expectation to be experts across a wide array of domains, from business requirements to intricate technical frameworks, grasp intricate structures within increasingly complex systems and deal with dependencies, concurrency and distribution. The cognitive load is overwhelming, leading to inefficiencies and errors.

Artifact proliferation: The SDLC generates a staggering number of artifacts, from strategy documents to deployment scripts. Managing and streamlining these artifacts toward successful delivery is resource-intensive and challenging.

Quality standards: Conformance to quality standards and intended architectures is only partially aided by frameworks and toolchains, leaving significant gaps that need to be manually bridged.

Time constraints: As project deadlines loom, time spent on designing and testing systems is frequently cut short, jeopardizing quality and increasing the risk of errors.

Legacy systems: Maintaining and evolving legacy systems remains a high-risk, manual effort. These systems are often critical to operations but are plagued by obsolescence and technical debt.

Carefully designed and implemented AI interventions can dramatically reduce the human toil in coping with every one of these challenges. By doing so, enterprises can maximize the value they realize from their application development and maintenance efforts—activities that consume an average of 45 percent of IT budgets, according to Gartner 2023 research.

This emphasis on code completion and the developer role alone obscures a larger opportunity.

At Publicis Sapient, we've discovered that leveraging AI across the entire SDLC, as well as across Strategy, Product, Experience, Engineering and Data & AI (SPEED) disciplines, can drive up to a 40 percent increase in productivity.

Yet, less than half of this gain comes from developer activities.

increase in productivity

STRATEGY
Competitive research, trend and behavioral analytics, hypothesis hacking

PRODUCT
Team casting, project initiation, intelligent planning, program reporting, backlog acceleration, hypothesis hacking

EXPERIENCE
UX research planning/execution, user journey analysis, UI prototyping, content generation, simulation & prediction

ENGINEERING
Code generation, bug detection & fixes, documentation, refactoring, test case creation

DATA & AI
Data cleansing, synthetic data generation, automated insight generation, model optimization, NLP queries

publicis sapient

Where does it come from? Significant potential lies in areas like strategic planning, design and release management—often overlooked due to the spotlight on tools like GitHub Copilot.

What are the top AI use cases in the software development lifecycle?

By focusing on the full SDLC from concept to production, we're not just speeding up code development; we're accelerating the entire process of delivering customer and business value, adapting swiftly to feedback and maximizing impact by trimming low-value features.

This holistic approach is crucial for truly agile and effective software development:

Strategy and planning: Competitive research, web and behavioral analytics

Gen AI productivity gains: 20%–40% reduction in time for identifying actionable trends

Design: Designs, reverse engineered code plans using repositories, backlog creation/ management, story sizing, sprint planning

Gen AI productivity gains: 20%–40% reduction in drawing IAs and designs

Coding: Generated code, integrations and data models

Gen AI productivity gains: 30%–50% reduction in net engineering time

Building, testing, verifying and integrating: Self-testing code, exhaustive scenarios and end-to-end testing

Gen AI productivity gains: 50%–60% reduction in code defects, overall time to test

Release and maintenance: Automated dashboards, insights, traffic and monitoring

Gen AI productivity gains: 20%–30% reduction in mean time to resolve and operational readiness

Source: Publicis Sapient & industry analysis, BCG, project to product state of the industry report 2023

	SAA USE CASES				
	20%–25%	10%–15%	10%–15%	20%–30%	30%–40%
SDLC stage and % of time spent	**Strategy and planning** *Competitive research, web and behavioral analytics*	**Design** *Designs, reverse engineered code plans using repos*	**Coding** *Generated code, integrations and data models*	**Build, test, verify, integrate** *Self-testing code, exhaustive scenarios, end-to-end testing*	**Release** *Automated dashboards, insights, traffic and monitoring*
Gen AI productivity gains	**20%–40%** *reduction in time for identifying actionable trends*	**20%–40%** *reduction in drawing IAs and designs*	**30%–50%** *reduction in net engineering time*	**50%–60%** *reduction in code defects, overall time to test*	**20%–30%** *reduction in mean time to resolve and operational readiness*

+10% to 15% increase in time due to complex user needs, additional governance, security controls, IP checks, data controls and human interventions

40% *increase in productivity and a reduction in time from idea to live*

Using lean value streams for optimization

To capture economic value, we apply lean thinking across the complete SDLC, combining value stream mapping and a catalog of targeted AI-enabled interventions. These are the same techniques that made the Toyota Production System famous—applied to the SDLC process.

When prioritizing where to start introducing AI into the workflows, it's essential to prioritize, plan and measure the impact of the interventions.

For example, in one set of measurements with GitHub Copilot, we see the highest gains with AI assistance during code creation in Java and React, with approximately 50 percent improvements, while with code optimization and defect fixing, gains are more modest, varying between 15 to 25 percent.

Finally, it is important to plan for some activities that take more time. For example, generating code might be a lot faster but we might spend a bit more time on ensuring explainability, tests and verification.

It provides insight only into tool usage and not the workflow in which it is being used.

Measuring and managing value, speed and quality improvements

In our experience, diligent measurement of speed and quality is essential for optimizing AI-enabled workflows.

Relying solely on AI tool analytics is insufficient; it's important to track detailed metrics across the entire SDLC to understand where productivity and quality improvements are truly occurring.

For example, at this time, the REST API for GitHub Copilot usage metrics provides only aggregated data, such as accepted code suggestions, which is insufficient to examine individual trends or those for each scrum team.

PS KnowHOW is our open-source metrics platform used to diligently measure throughout the SDLC. It gives us real-time visibility into cycle time, quality and value metrics, including the ability to measure the impact of AI interventions.

Cross-discipline collaboration and alignment

AI benefits more than just developers. A transformative benefit of AI is its capability to connect, cross-analyze and generate artifacts throughout the SDLC for different roles, including strategists, product managers, agile program managers, experience designers and data scientists, which can further strengthen collaboration and innovation.
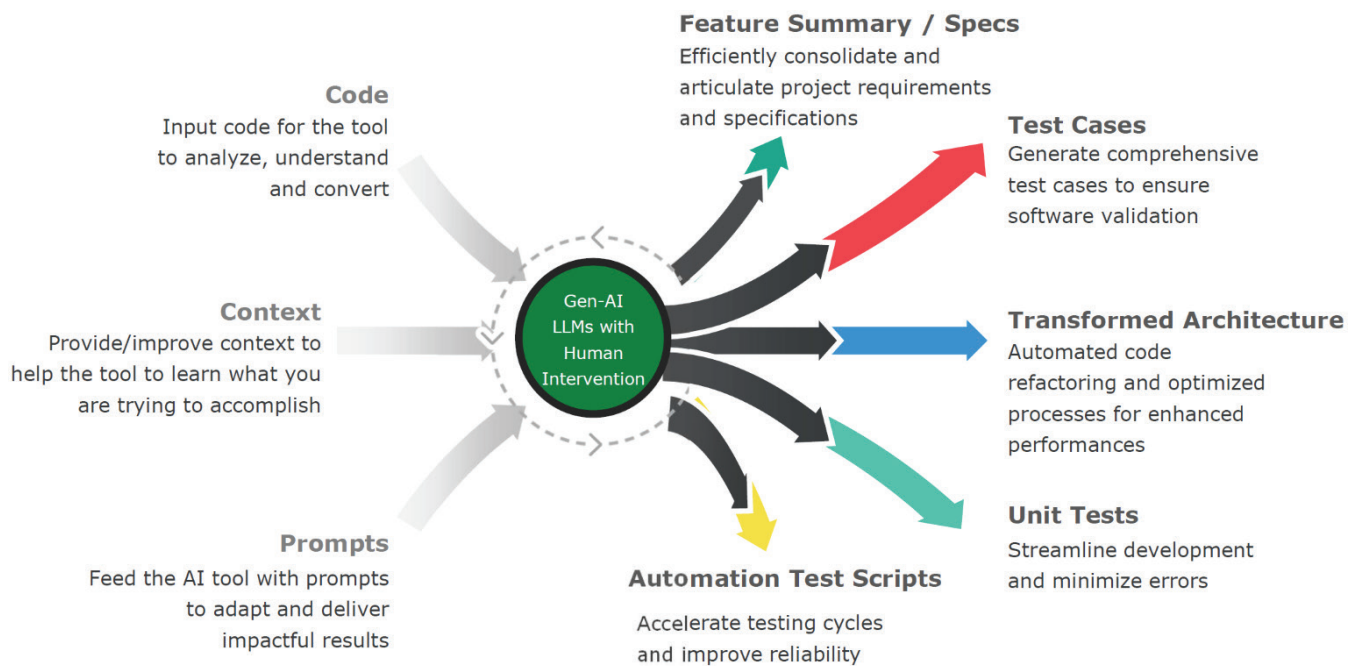
AI can help maintain alignment between strategy, design and implementation, ensuring that the final product adheres closely to original business and architectural intentions—a critical advantage in complex systems where manual oversight often falls short.

Real-time feedback and mentoring

AI can also offer real-time feedback and mentoring, help practitioners refine their skills and avoid common mistakes, create first drafts and review work. For example, solution product managers can leverage AI to generate or critique backlogs and test cases, while developers can utilize AI as a peer programmer, enhancing both individual and team productivity.

Legacy modernization

It's not just new application development and routine maintenance that stand to gain. Many large enterprises, particularly those with decades of history, struggle with outdated legacy systems. Simply lifting and shifting to the cloud is not a viable option for many of these systems. Modernizing them to cloud-native technology stacks has, until now, been anything but trivial.

Code
Input code for the tool to analyze, understand and convert

Context
Provide/improve context to help the tool to learn what you are trying to accomplish

Prompts
Feed the AI tool with prompts to adapt and deliver impactful results

Gen-AI LLMs with Human Intervention

Feature Summary / Specs
Efficiently consolidate and articulate project requirements and specifications

Test Cases
Generate comprehensive test cases to ensure software validation

Transformed Architecture
Automated code refactoring and optimized processes for enhanced performances

Unit Tests
Streamline development and minimize errors

Automation Test Scripts
Accelerate testing cycles and improve reliability

We've conducted over 100 end-to-end legacy transformation experiments using our PS Monarch AI-powered modernization accelerator, achieving:

- *Greater than a 50 percent reduction in modernization costs*
- *50 percent fewer defects compared to traditional methods*
- *Up to a 70 percent reduction in cycle times*

03 The need for specialized tools and platforms for AI-assisted software development

The potential for AI to transform software development is immense, but how can enterprises truly harness this power? Is it as simple as deploying a code completion tool or offering a secure version of a leading LLM chatbot with some user training?

While those tools are logical to start with, our experience is that they only scratch the surface and fail to maximize the achievable gains in value, productivity and quality.

Why? Special Forces teams, such as Navy SEALs or Green Berets, don't use standard-issue equipment when embarking on critical missions. They rely on customized, mission-specific tools—whether it's modified weapons, specialized communication devices or advanced surveillance technology. These tools are meticulously designed and adapted to meet the exact demands of their mission, allowing them to operate with precision and effectiveness in high-stakes environments.

Tasks inside the frontier

Tasks outside the frontier

Tasks on the frontier
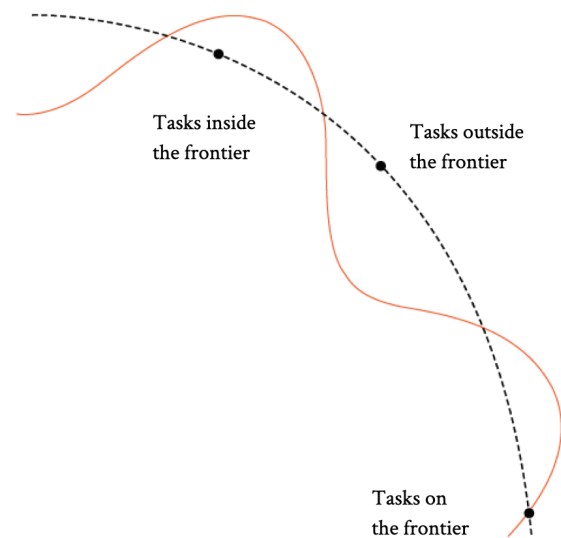
The "jagged frontier" of AI capabilities

To appreciate why this relates to AI in the SDLC, it's essential to grasp when LLMs like GPT-4 benefit knowledge workers and when they are harmful. The concept of the "jagged frontier," introduced in a Harvard Business School (HBS) and Boston Consulting Group (BCG) paper, offers a vivid metaphor for understanding what happens when chat-based AI tools are offered to skilled professionals.

Picture this:

- A fortress wall with towers and battlements of varying heights, representing AI's uneven proficiency across different tasks

- Within the fortress walls, AI excels, leading to significant productivity and quality gains for the BCG consultants who were studied

- Beyond those walls, however, AI struggles and can provide incorrect or misleading information, leading to the BCG consultants being 19 percent less likely to produce correct solutions compared to those not using AI

This jagged frontier serves as a cautionary tale for those relying on out-of-the-box LLM chatbots or code completion tools. The metaphor reminds us that while AI can greatly assist with tasks it's well suited for, it can also be harmful when applied to tasks beyond its current capabilities.

The power of fine-tuned AI

Unfortunately, for the highest-value software development use cases, general-purpose LLMs lack the specific domain and enterprise context required to be genuinely effective in enterprise environments.

In effect, many valuable SDLC use cases are outside the castle walls—beyond the jagged frontier. In software development, the real power of AI emerges when LLMs are fine-tuned with unique enterprise knowledge, use case-specific context and by chaining custom task-specific agents and guardrails. That fine-tuning brings the high-value use cases within the castle walls.
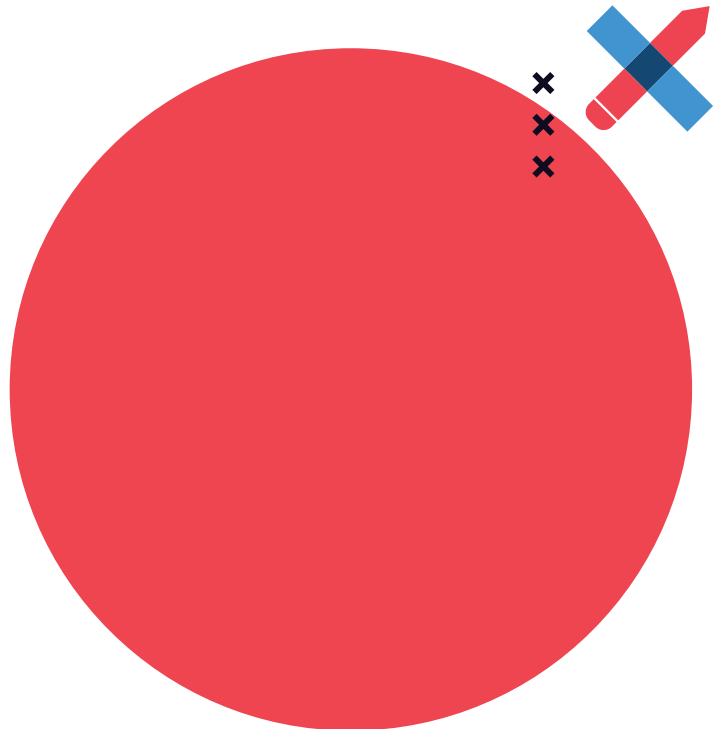
Why prompt engineering is not enough

Fine-tuning LLMs with solution patterns, enterprise context and guardrails isn't just a nice-to-have—it's crucial for generating enterprise-ready AI outputs. This is simply not possible today with general-purpose LLMs. Some may argue that well-crafted AI prompts can contain all the enterprise context needed to allow the AI to make a useful response. Our experience has shown that while large context windows in modern LLMs allow for generously long prompts, attempting to include all the context in the prompt alone can lead to surprisingly inconsistent results, more frequent hallucinations and diluted relevance. Simply put, relying solely on prompt engineering or vanilla code completion without deeper customization will not, as of this writing, yield the best AI outputs for enterprise software development.

How specialized tools and platforms help

To address this gap, we've adopted a **new** approach: building task-specific and workflow-enhanced acceleration products that include fine-tuned models enriched with use case, domain, workflow and enterprise context and pre-authored guardrails, supplemented by curated prompt libraries.
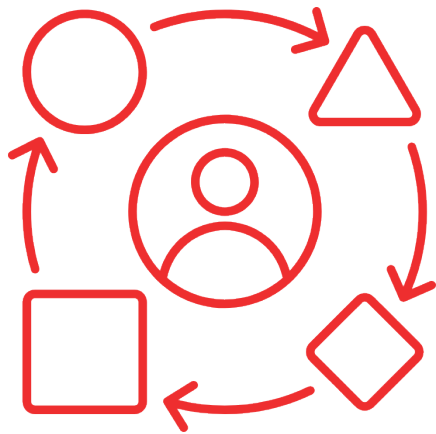
Enterprise software development challenges require more than just off-the-shelf solutions. General-purpose LLMs, while immensely powerful, lack enterprise, task-specific and workflow context, and that context cannot be solved by prompt engineering alone.

Successful AI-assisted software development involves developing task-specific and workflow-enhanced acceleration products, much like how Special Forces customize their gear for each mission. These products are built on fine-tuned AI models enriched with use case, domain, workflow and enterprise context. By incorporating guardrails, response templates and curated prompt libraries, the software development team using these fine-tuned AI tools can be confident that the outputs meet best practices for the enterprise solution archetype they are working on.

Examples of fine-tuned AI in action

For example, an AI trained on your visual design system can be a game changer for experience designers and front-end developers, ensuring that AI outputs align perfectly with your brand and front-end code standards. Similarly, an AI model familiar with your enterprise architecture and APIs can ensure that its outputs make correct use of your IT foundation, driving efficiency and consistency as it produces and inspects code and test cases. An AI model trained on the nuances of moving an Adobe Experience Manager implementation to Adobe's cloud service can now assist with such migrations repeatedly and accurately, ensuring its outputs correctly use the architectural patterns consistent with enterprise standards.

Case in point: Amazon used a tuned version of its AI product called Q to upgrade their foundation software applications to Java 17. They observed that by curating the migration pattern and tools with Q, the average time to upgrade an application plummeted from what's typically 50 developer days to just a few hours.

It's crucial to invest in expert AI toolmakers to **build the tooling specific for each archetype**. The toolmakers ensure that the domain expertise and nuances of the archetype and technology stack are built directly into the AI toolset, thus ensuring the relevance and safety of AI outputs. Each archetype has its own AI toolset, which is managed as a product, to maximize the productivity and quality of the software development teams building solutions of that type.

Adaptability is the key to success

In both the battlefield and the business world, adaptability to the environment and mission context is crucial. The key to building the right AI tools for enterprise software development is identifying the different archetypes or solution patterns where the AI-powered toolset must be precision-built to ensure maximum productivity and quality.

04 Issues, risks and mitigations with AI-assisted software development

Many books will be written over the next few years on the risks of generative AI. When using AI for software development, we can narrow our risk surface area with a product-driven approach to AI tools. The product teams responsible for engineering the precision tools must make sound decisions to address the following issues and safeguard the enterprise:

Are AI outputs explainable?

Explainable AI is critical because the less we make generative AI a black box and the more we understand not just the output but why it chose that output, the better. This is especially true when we use AI for critical enterprise applications.

For example, AI may produce code that works, but developers can't fully understand how or why it was generated that way. AI tools might flag potential bugs without clearly explaining the reasoning behind the detection. AI-assisted design tools could make choices that designers can't easily justify or explain to stakeholders. Similarly, AI-generated requirements may not provide clear rationales to product managers for why certain scenarios were chosen.

This risk can be mitigated by various techniques, such as asking the AI to explain different sections of its outputs, comparing the output of one AI model with another and asking for explanations of key differences or using chain-of-thought prompting.

Chain-of-thought prompting encourages AI models to break down complex problems into a series of intermediate steps, mimicking human-like reasoning. This approach involves providing exemples that demonstrate step-by-step reasoning, thus guiding the model to generate similar reasoning chains for new tasks. This enables the model to articulate its thought process, leading to more transparent and interpretable outputs.

Advancing AI explainability: How OpenAI's o1-preview model enhances reasoning and accuracy

OpenAI's latest o1-preview model achieves superior reasoning through a combination of reinforcement learning and chain-of-thought reasoning. Reinforcement learning enables the model to continuously refine its problem-solving strategies as it explores constraints, learning from mistakes and adjusting its approach to deliver more accurate and logical outcomes. By explicitly mapping out its reasoning process, o1 not only explains its output but also identifies its own potential errors and addresses them, significantly improving the likelihood of arriving at the correct solution—much like how humans reduce mistakes when we carefully outline our thinking. We believe o1 is a significant step forward in improving AI explainability, particularly for our SDLC use cases.

Security and disclosure

All enterprises want their secrets to be, well, secret. For many enterprises, it is sufficient to ensure that the enterprise legal agreements with the LLM API provider, like Microsoft, Google, OpenAI or Anthropic, have the right covenants in place to ensure confidentiality and prohibit data leakage, particularly to address the concern that the enterprise prompts or context data will be used to train the external base models. These agreements, in concert with a gateway in the enterprise that scans for and masks sensitive data, like in our own PSChat, might be sufficient.

With some of our clients who work with classified material or high-stakes intellectual secrets, we go the extra mile to ensure that sensitive work products never leave the enterprise by hosting LLMs, such as Meta's CodeLlama or Mistral, in their own environment. For these clients and in regulated industries like financial services, healthcare and sensitive government sectors, establishing common enterprise foundations that control and monitor access to, enforce security policies for and provide usage visibility of generative AI resources is key. Our PS GenFoundry and PS Bodhi frameworks package and provide these solutions.

This is especially applicable to AI-assisted software development. The humans who are guiding and inspecting the AI work require more expertise, not less, whether those humans are business strategists, product managers, experience designers, software engineers or data professionals. They must be curious, skilled at problem decomposition and experts at double-checking the AI so they can take responsibility for the output of the AI and the outcomes of the solution.

Concerns about disclosure are not always applicable. For example, if you're trying to create a web reporting application, you can instruct an AI with a prompt like, "Generate a web app that has tables and checkboxes, and format the data as follows." This allows you to avoid disclosing any proprietary information while still saving a lot of development effort, as neither the prompt nor the output contains sensitive secrets. However, executives must train employees to discern what is confidential information and establish suitable compliance guardrails.
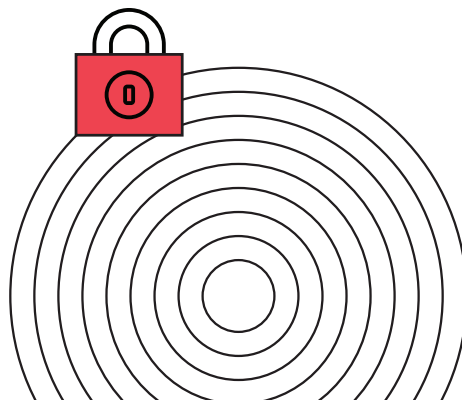
How to pick use cases that minimize risk

With rapid advancements in AI, there's a temptation to apply it everywhere all at once. IT leaders must evaluate risks by considering the potential impact of AI errors and the ease of detecting and fixing the errors, should they be present.

Just as a risk manager evaluates the likelihood, impact and mitigations for each risk, leaders must evaluate each use case by asking two questions:

a. For this use case, if the AI were to make an error, how large would the impact of the error be?

b. For this use case, if the AI were to make an error, how easy would it be for my software development team to find the error?

Using these questions, it would be apparent that LLMs today are less useful in high-impact, harder-to-inspect tasks like verifying system architectures compared to labor-intensive but lower-consequence, easier-to-inspect tasks like documentation or test case generation.

How AI errors can be mitigated

We must design our AI-assisted workflows with the new checks in the process, commensurate with the use case. For example, with code generation, the benchmark is perfection—no IT leader would knowingly release buggy code into production.

Human developers make mistakes. Therefore, we invented test-driven development and static analysis to assist human developers, and we integrated these tools into our workflows and automation to reduce human error.

Today's LLMs are incredibly capable and sophisticated token-guessing machines. They too will make mistakes, even if those mistakes become increasingly rare as the LLMs improve. Therefore, we must take the same approach and design our workflows to nudge the AI in the right direction and verify its output using a combination of human review, adversarial AI (where one AI is used to inspect and challenge the generating AI) and other verification techniques. Trust relies on empirical data and ensuring we have rigorous testing processes and human oversight.
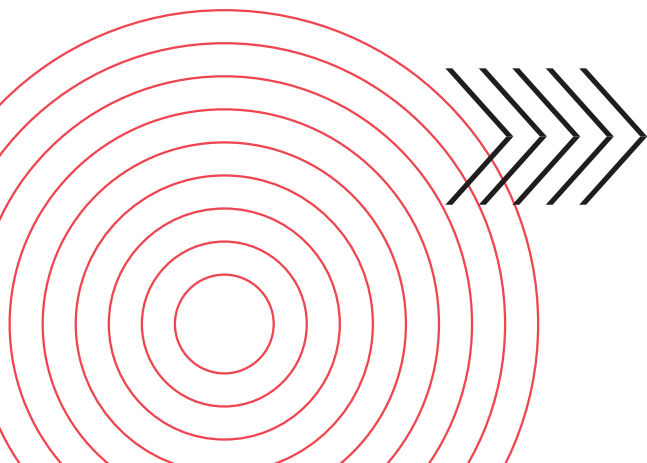
The risk of infringing on others' intellectual property

Choose AI tools from established model providers that have trained their models on code they have rights to use and offer indemnification or legal protection for AI outputs. Keep abreast of ongoing lawsuits and legal decisions regarding LLMs and copyright, as this area of law is still evolving.

The AI skills needed to mitigate new AI risks

For every AI-augmented task in the SDLC, a human employee must be in the driver's seat. It is a person who points a generative AI tool at the specific kind of problem that they want to solve, provides assistance to generative AI in terms of structuring the problem into smaller, solvable pieces and, importantly, audits, reviews and sanity-checks the results. When employing specialized tools and platforms, the product team that builds the AI toolset builds in guardrails to increase safety, but the software development team must ensure that the AI outputs are fit for purpose.

The biggest risk with the use of AI is inadequate human skills. For example, in human pair programming when an experienced developer is paired with a junior one, there is a risk of the "Watch the Master" phenomenon, where the junior developer becomes too passive.

This is especially applicable to AI-assisted software development. The humans who are guiding and inspecting the AI work require more expertise, not less, whether those humans are business strategists, product managers, experience designers, software engineers or data professionals. They must be curious, skilled at problem decomposition and experts at double-checking the AI so they can take responsibility for the output of the AI and the outcomes of the solution.

Proactive training must equip the team to rigorously scrutinize and verify AI outputs, regardless of how credible and complete they may appear to be. There are also use cases where generative AI, due to biases in the training data, presents some subtle issues.

It's important to know where to apply extra diligence in both application and verification.

05 Preparing for what's next: future trends in AI-assisted software development

As we navigate the future of AI-assisted software development, the role of enterprise IT is more critical than ever. The opportunity is not just about reducing costs or replacing human effort but about unlocking new opportunities for growth and innovation.

To prepare for the future of AI-assisted software development, business and technology leaders must invent the future.

Inventing the future and Jevons paradox

In the future, we can assume that humans will build more software. This article on Jevons paradox and its implications for software development explains why this is inevitable. It describes the counterintuitive situation where technological advancements that make a resource more efficient actually lead to greater overall consumption of that resource, driven by increased demand, rather than a reduction in its use.

For example, improvements in fuel efficiency led to more driving and thus higher overall fuel consumption. Advances in storage technology lower the cost per gigabyte, but the ease of storing large amounts of data encourages more data creation and retention, leading to increased storage infrastructure and resources to power and manage it. If Jevons paradox holds for AI-enabled software development, as it has with countless other technologies, those

businesses that redouble their investments in digital business transformation will have far better odds of maintaining and growing their competitive edge.
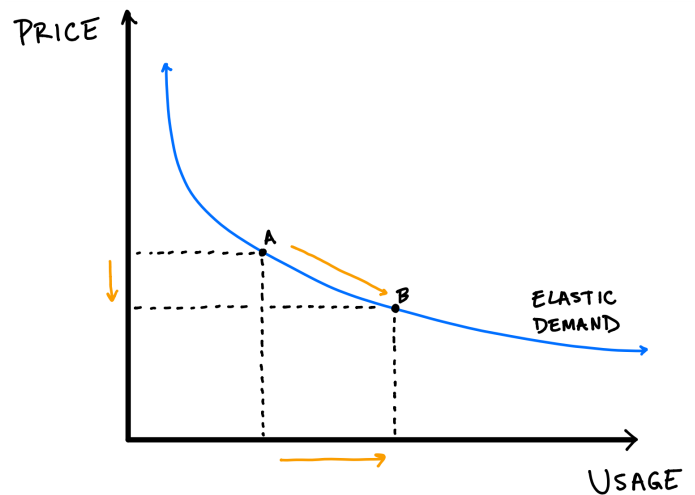

Source: Matt Rickard via Substack

This brings us full circle to Marc Andreessen's famous observation that "software is eating the world." While his idea captures the imperative for digital business transformation, it raises a crucial question: If software is indeed taking over, why does it seem to take so long to reach its full potential? More importantly, if software were truly to dominate, what would drive this transformation, and what would our world look like? The opportunity to answer these questions is finally available to every business executive, not just those with the resources of digital natives.

Taking a leadership position will require a focus on:

1) Transforming IT capabilities
2) Moving from general contractors to investors
3) Exploiting proprietary assets
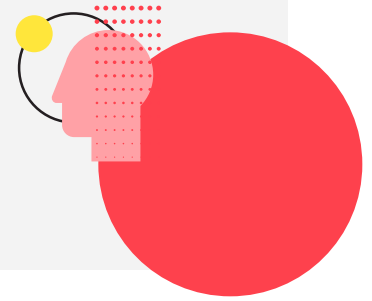
1) Transforming IT capabilities

The greatest lift in creating differentiated transformation capabilities lies in skills, talent and organizational transformation. The professionals in our organizations who conceive and build software across Strategy, Product, Experience, Engineering and Data & AI (SPEED) must embrace AI as part of how they work—transforming into cyborgs that seamlessly blend their own intelligence with AI to multiply their efficiency and effectiveness.

This is not just an individual endeavor, but rather one that leaders thoughtfully design, incorporating new roles to support evolving workflows. For example, the myriad technical specialist skill sets in IT evolve into a pool of versatile AI engineers who seamlessly use AI to build solutions across multiple technologies. These AI engineers are supported by a much smaller cohort of deep AI software development product specialists who design workflows and build the tools that the multiskilled AI engineers use.

IT leadership must reimagine and redesign their capabilities, starting with workflow, organization and technology enablers to build this future.

2) Moving from general contractors to inventors

IT needs to focus on more than just creating, testing and launching software, which is often referred to as moving from "backlog to live." With the speed at which software can now be developed, the real challenge in creating unique value will shift. Instead of just focusing on writing and deploying code, companies will need to pay more attention to how they come up with new ideas, create strategies and design experiences.

Just as a chain is only as strong as its weakest link, the software development process is only as fast and efficient as its slowest parts. IT leaders must rethink and improve the entire process, from coming up with ideas to managing them and finally getting them into production. In essence, this is about realizing the full promise of digital business transformation through product engineering, of which software development is a part.

3) Exploiting proprietary assets

In May 2023, an internal document from a Google employee was leaked and published online in a memo discussing the lack of competitive advantage in LLMs. The memo argued that neither Google nor OpenAI had a sustainable competitive advantage (or "moat") in LLMs compared to open-source efforts using internet datasets. It was controversial when published but has been seen by some as prescient given subsequent developments in open-source AI models. We can already see the lack of sustained differentiation play out in the rapid race for model supremacy amongst OpenAI, Google, Meta, Anthropic and others. In contrast, large enterprises have a significant moat—proprietary corporate data, knowledge assets and expertise. These can be used to build specific tools fine-tuned for creating accretive enterprise value. Investing in data curation and model fine-tuning will yield significant acceleration compared to using public models alone. Training employees on prompting for the unique capabilities of these fine-tuned models will yield further differentiation.

Conclusion:
The new AI frontier in software development

As we look ahead to the evolving landscape of software development, it's clear that AI-assisted innovation is set to redefine the way we work. The strategies we've discussed highlight how AI can drive efficiency, improve quality and support growth. Rather than being just another tool, AI is becoming a crucial element in digital transformation. Now is the time for leaders to explore this potential, guide their organizations through change and unlock the new possibilities that AI brings to the future of software development.

How Publicis Sapient can help

A successful AI approach is built on digital business transformation—and DBT is at our core. Our team helps prepare for AI and generative AI implementation, evaluate and prioritize use cases and execute the right strategy for your business.

Recognized as a 2023 Market Leader in Generative Enterprise Services by HFS Research, our teams have experience designing and building innovative AI solutions. Together, we can unlock the full value of AI with an ethical and sustainable approach.

Discover *Sapient AI*, Publicis Sapient's ethos for purpose-driven and scalable AI solutions.

Sapient Slingshot

Sapient Slingshot is our proprietary AI toolset designed to enhance software development across all SDLC stages. It provides tools for code generation, testing, deployment and more, backed by our industry-specific data and prompt library. Beyond the tools, our experienced team supports implementation and customization to meet your needs, helping engineers, product managers, scrum masters and designers deliver better, faster and more cost-effective solutions.

AI application modernization

Boost operational efficiency and enable innovation by modernizing your legacy systems with the help of AI. Accelerate developer productivity, enabling faster code transition, documentation and automated testing for scalable growth.

AI custom application development

Create differentiated and dynamic experiences through application development using AI. Code, test and deploy at an accelerated rate, allowing you to focus on identifying the features your customers want, while continuously optimizing and innovating.

AI MarTech transformation

Drive richer customer connections and efficiency within your marketing organization, enabled by a leading MarTech stack infused with AI.

AI test automation

Enhance software quality and accelerate release cycles with AI-powered test automation. Streamline test creation, execution and maintenance, enabling rapid identification and resolution of issues.

publicis
sapient

Publicis Sapient is a digital transformation partner helping established organizations get to their future, digitally enabled state, both in the way they work and the way they serve their customers. We help unlock value through a startup mindset and modern methods, fusing strategy, consulting and customer experience with agile engineering and problem-solving creativity. As digital pioneers with 20,000 people and 53 offices around the globe, our experience spanning technology, data sciences, consulting and customer obsession—combined with our culture of curiosity and relentlessness—enables us to accelerate our clients' businesses through designing the products and services their customers truly value. Publicis Sapient is the digital business transformation hub of Publicis Groupe. For more information, visit **publicissapient.com**.