QuantSpark

Harnessing AI to Accelerate Innovation

QuantSpark's strategic approach to evaluating and implementing AI coding tools

Executive Summary

Making sense of the AI tools on offer:
This Insight Report gives a useful overview for companies wanting to leverage the latest AI pair programming tools.

Evaluation and Implementation:
Companies should prioritise a tool's compatibility with their tech stack, its training data, compliance and security features.

Taking control of AI:
The promise of AI coding tools should not be underestimated, but success hinges on alignment with company needs, adequate training, and human oversight. They complement, not replace, human developers.

Introduction

AI coding tools leverage innovations in large language models to assist software engineers and developers in writing, testing, and documenting code.

These tools can analyse a developer's coding style and patterns to provide suggestions for completing lines of code or entire functions. The tools are trained on massive datasets of code to gain an understanding of syntax, logic, and structure for various programming languages.

AI coding tools have the potential to generate measurable benefits like increased productivity, faster time to market, and higher code quality

As more companies look to gain a competitive advantage through software and technology, the demand for engineers continues to outstrip supply. AI coding tools are a way to improve developer productivity and help address this shortage. They can save developers time, reduce errors, and allow them to focus on higher-level work.

For companies evaluating these tools, it is important to consider not just the benefits but also the limitations and risks. Success will depend on factors like the task, developer skill level, and the programming languages and integrated development environments (IDE) supported for how well the tool integrates into existing workflows.

There are also concerns about models that are trained on proprietary code and the intellectual property risks from using such tools.

The interest in this space can be justified and with the right approach, AI coding tools have the potential to generate measurable benefits like increased productivity, faster time to market, and higher code quality.

Benefits of AI Coding Tools

Explaining and summarising code:
AI tools can quickly analyse large codebases and provide overviews of how systems work. This helps onboard new developers faster.

Writing boilerplate code:
For repetitive coding tasks like creating classes, functions, and tests, AI tools can generate the necessary boilerplate code. Developers simply fill in the details.

Coding in unfamiliar languages:
For developers working in a new programming language, AI tools provide suggestions based on their knowledge of syntax and best practices. This makes the learning curve smoother.

Detecting errors and security risks:
AI tools can analyse code, assist with debugging, and detect issues like syntax errors, security vulnerabilities, and non-compliant code. They provide alerts so developers can address problems quickly.

Writing documentation:
By understanding code, AI tools can generate initial documentation based on comments, function names, and logic. Developers then simply review and improve the documentation.

92%
of developers say they use AI coding tools at work [1]

70%
of the developers we surveyed say they already see significant benefits whenusing AI coding tools [2]

55%
reduction in time for programming tasks [3]

The Factors for Success

The outcomes and benefits of AI coding tools depend on several factors:

Type of task:
The more repetitive and rules-based the task, the more AI tools can assist. They are less capable with highly complex, abstract coding tasks.

Developer skill level:
Less experienced developers will likely gain more from AI coding tools by learning best practices and proper syntax. Highly skilled developers may find less utility in the tools, which may assist more with routine and repetitive task.

Technical factors:
The languages, frameworks, and IDEs supported by a tool determine which developers and environments can use it. More flexibility and compatibility mean wider usage.

Other factors:
Formal training, organisational support, and the performance of the tools themselves also impact success. Lack of training and support will limit adoption. Poor AI coding tools must provide recommendations with minimal latency to avoid disrupting developer workflow. Higher latency impacts productivity and poor tool performance can frustrate developers.

Source:

(1) github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developer-experience/

(2) github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developer-experience/

(3) github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/
*might be an overestimation, productivity gain depends on a number of factors

How to Evaluate AI Coding Tools

There are several factors companies should consider when evaluating different AI coding tools:

General Information

Year launched and status:
Is the tool in production or still in beta?

More established tools are likely to be more stable and feature-rich.

Open-source vs proprietary:
Open-source tools can be freely adapted to a company's needs but may lack support. Proprietary tools often have more features but can be expensive.

Technical Features

Data trained on:
The data used to train an AI coding tool significantly impacts its effectiveness and areas of proficiency. Tools trained on larger datasets or with code similar to a company's own can provide more accurate recommendations.

Fine-tuning capabilities:
The ability to adapt a tool to a company's own codebase through retraining the model or other means allows for a customised experience. Without fine-tuning, a tool may provide generic recommendations that do not align with internal best practices.

Self-hosting options:
Tools that can be deployed locally provide more control and security for a company's data and IP. Cloud-based tools may be more convenient but also present risks around data privacy and protection of proprietary code.

Workflow integration:
AI coding tools should integrate seamlessly with the platforms and tools developers already use. For maximum benefit, a tool should support the languages, frameworks, and IDEs used in a company. The more of a tech stack a tool can support, the more useful it will be to a wider range of developers and projects. Disrupting established workflows will hamper adoption and productivity.

Usability and Impact

Learning curve:
Tools with an intuitive, easy-to-learn interface will be adopted more quickly by developers. Complex tools require more ramp-up time.

Best use cases:
Different tools have strengths for particular types of tasks. Companies should evaluate tools based on their ability to assist with the organisation's most common and impactful coding needs. The highest-value use cases may depend on a company's tech stack, industry, developer experience level, and other factors.

Measure the benefits:
Companies should measure metrics before and after implementation to quantify impact. Organisations that offer AI coding tools present a range of statistics for their benefits; some might inflate the uplift more than what an organisation actually experiences.

Performance and latency:
Developers need coding high quality suggestions with minimal latency to achieve maximum productivity. Tools should provide real-time recommendations with no disruption.

Support and community:
Dedicated support from tool developers and an active community of users to provide guidance and share best practices is important, especially when first implementing a tool.

Cost and Licensing

Pricing model:
Pricing models include free open-source, monthly subscription, and per-seat licensing. Companies should choose a model that fits their budget and needs.

Licensing terms:
For proprietary tools, licensing terms govern how the software can be used, modified, and distributed. Companies should ensure terms align with their goals.

Key Considerations: IP, Privacy, and Quality

It is important to consider these points:

IP and legal risks:
AI coding tools trained on proprietary, copyrighted code from other companies could present IP and legal issues. If a tool's recommendations are substantially similar to code from its training data, companies relying on the tool may face infringement claims. Companies should evaluate tools for IP contamination risk and negotiate indemnification clauses with tool vendors in case issues arise from their recommendations. There is currently legal uncertainty around some models which companies must consider.

Privacy and security:
AI tools may require access to proprietary code and data. Companies must ensure tools have proper security and governance to prevent IP or data leakage with appropriate privacy controls and anonymisation.

Quality of output:
The code suggested by AI tools should meet or exceed organisational standards for efficiency, readability, security, and maintainability. Human review is still often required.

Flexibility and compatibility:
Tools that support more languages, frameworks, and platforms can be used on more projects. More flexibility increases the potential for benefit.

Evaluation Checklist

Criteria	Considerations
Open-source vs proprietary	Are we looking to customise or for an off-the-shelf solution?
Data trained on	Does it include proprietary code?
Fine-tuning capabilities	Can we fine-tune to our needs?
Self-hosting options	How important is security? Can we maintain this?
Workflow integrations	What programming languages and IDEs are supported?
Ease of use	How's the UX? Are there good documentation and support?
Performance	Is the data trained on good-quality? How's the latency and output for our use cases?
Cost & licensing	Do the pricing model and licensing terms match our needs?

Implementing AI Coding Tools: Best Practices

Once a company has evaluated and selected an AI coding tool aligned with their needs, the rollout across the organisation requires careful planning to achieve maximum benefit. Key steps include:

Start small with a pilot

As part of the assessment and evaluation of tooling, identify which teams might benefit most to help decide where to start the pilot. Deploy the tool with a single team or on a targeted project to measure performance and gain user feedback. Look for meaningful metrics tied to KPIs like productivity, time to completion, and error rates. Make adjustments before expanding usage.

Provide hands-on training

Require developers to complete interactive training to fully understand the tool's capabilities, limits, and proper application. In-person workshops and demos led by vendor experts or internal pioneers are most effective. Developers should feel comfortable with how and when to leverage the tool.

Make learning resources readily available

Give developers easy access to documentation, video tutorials, and support options to facilitate problem-solving. Internal collaboration channels where early users can share their experiences also help.

Regularly monitor KPIs and user sentiment

Continue tracking the metrics used during the initial pilot to ensure the desired productivity and efficiency gains. Also survey users periodically on their experience, challenges encountered, and suggested improvements. Monitor user feedback and address any concerns quickly.

Designate internal champions and supporters

Identify influential developers who are enthusiastic about the tool to provide peer support and mentorship for other users. They can also relay feedback to executives and vendors to guide enhancements and additional training or resources.

Review policies and governance

Assess how tool usage may impact areas like data privacy, security, and compliance. Update documentation, standards, and controls as needed to account for any new areas of risk. Changes may need to be made thoughtfully before proceeding to the next phase of rollout.

Expand thoughtfully over time

Take a phased approach to rolling out the tool across teams and projects. Move to the next group once current users have adopted the technology enthusiastically, experienced the expected benefits, and can help mentor new users. Phasing the rollout in this way maximises the probability of success at each stage.

With the right strategy and guidance, AI coding tools can deliver valuable benefits to organisations.

But their impact depends on how well companies invest in onboarding developers, address risks, and make the effort to gain buy-in through support, feedback, and continual optimisation.

Overall, start slow to go fast, focusing on sustainability over speed of rollout. The tools can then become lasting multipliers of human capability rather than novel experiments.

Key Takeaways

TAI coding tools show significant promise in helping companies to accelerate development of digital capabilities. With an expanding set of options available, companies looking to benefit from these tools should:

Evaluate tools based on their own tech stack, workflows, and coding tasks. The best tools align closely with existing systems and processes

Measure productivity and key metrics before and after implementation to quantify the impact. Look for meaningful gains in productivity, reduced errors, and time to market

Choose a **flexible** tool that can adapt to a company's needs over time. Consider both proprietary and open-source options, depending on requirements

Provide training to help developers take full advantage of the tools. Make learning and onboarding resources available

Continually **monitor performance** and adjust gain the most benefit. The tools and their models should evolve along with a company's needs

With the right tool and approach, AI coding technology has the potential to transform software development. But companies must go in with realistic expectations about the current capabilities and limitations of these tools. They are currently designed to augment human developers, not replace them. When combined with human judgment and oversight, AI coding tools can help propel companies into a new era of rapid digital innovation. But they require partnership, not automation alone.

QuantSpark

Authors

Adam Hadley
CEO & Founder
adam.hadley@quantspark.com

James Butcher
ML Innovation Lead
james.butcher@quantspark.com

Contact

contact@quantspark.com

+44 (0)20 8075 7677

230 Blackfriars Rd
London
SE1 8NW