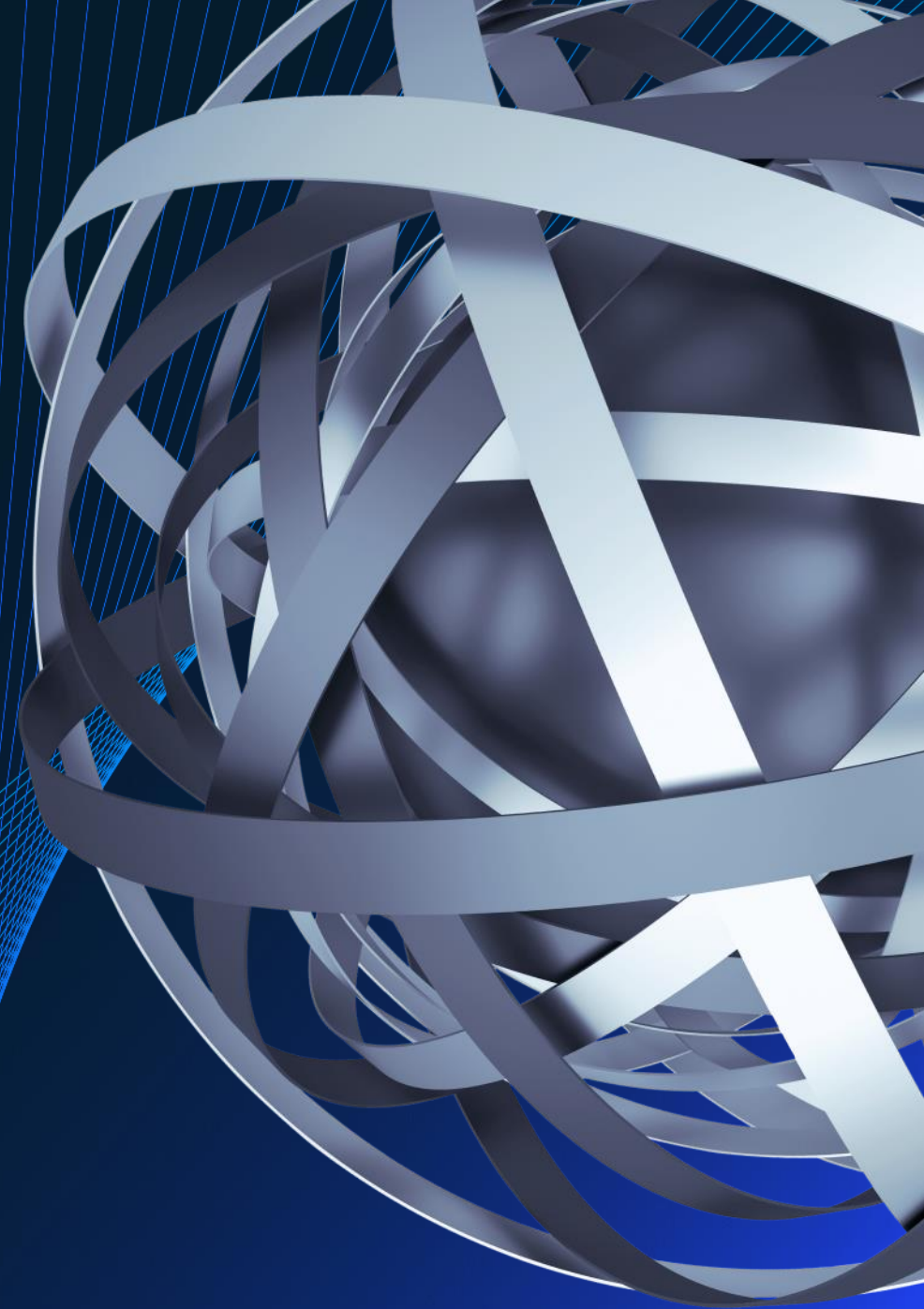


McKinsey
& Company

McKinsey Technology Trends Outlook 2022

**Next-generation
software development**

August 2022



What is this trend about?

The next generation of software development involves tooling that aids in the development of software applications, improving processes and software quality across each stage of the software development life cycle, including AI-enabled development and testing, as well as low-code/no-code tools



Technology or tool kit

Low-code/no-code platforms

Graphical user interface (GUI)-based platforms for nondevelopers to use in building apps

Infrastructure-as-code

Configuration templates to provision infrastructure for applications using Terraform, Ansible, etc

Microservices and APIs

Self-contained modular pieces of code that can be assembled into larger applications

AI “pair programmer”

Code recommendations based on context from input code or natural language

AI-based testing

Automated unit and performance testing to reduce developer time spent on testing

Automated code review

Automated software checks of source code through AI or predefined rules

Why should leaders pay attention?

Developers will focus more on the capabilities their applications would enable than on the details of building the apps

Growth in market and adoption



Greater adoption



~70%

Share of new application development that will leverage low-code/no-code by 2025 (vs <25% in 2020)



Growth in market size



~21%

Growth in size of market for software development, **CAGR for 2021–26**, reaching ~\$600 million by 2026

Augmented capabilities



Faster development



Up to **~90%**

Reduction in development time due to low-code/no-code applications



Faster deployment



~2x

Increase in deployment speed reported by ~60% of developers, driven by practices such as continuous integration and continuous delivery (CI/CD)

10101
01010
10101

Faster code testing



~37%

Share of respondents saying they **use AI and ML to test better** and faster



Reduced resolution time



<1 day

Time to resolve configuration issues reported by ~75% of companies with automated infrastructure-as-code security testing

As repetitive tasks become automated and resource requirements to build digital products decrease, developers will **focus on adding new, innovative features**

Many methods, including CI/CD and infrastructure-as-code, will **benefit from cloud migration** and accelerate this transition

Why are the technologies interesting, compared with what already exists?

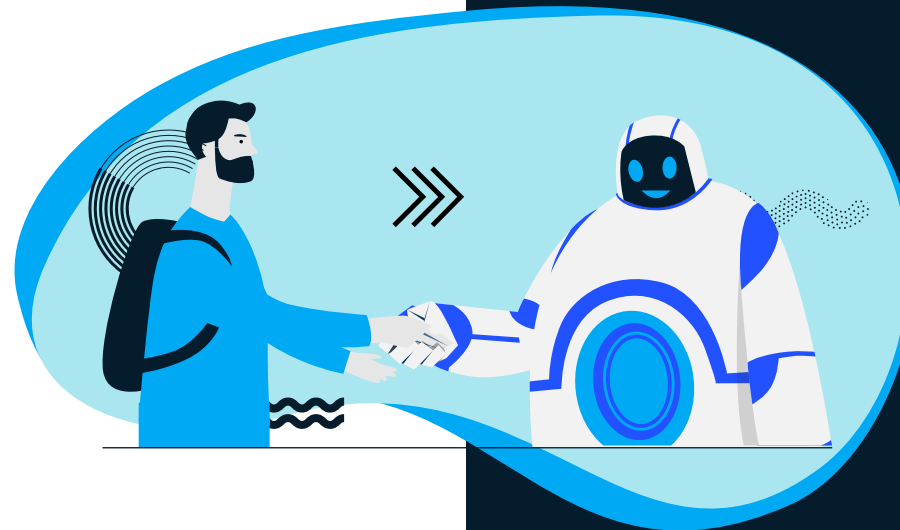
From manual, time-intensive work flows and techniques ...

Reliance on dedicated developers to participate in every step of the development cycle, from planning to maintenance, contributes to higher costs and talent gaps

Manual infrastructure configuration and monitoring involve high mean time to restore (MTTR), security risks, and task repetition, leading to inefficient resource utilization

Developers working together to write code as 'pair programmers' on the same workstation expend a high number of person-hours to build the program

Development cycles are slow because teams experience interruptions, code has more defects, and time is spent on manual tasks



... to automated, simplified, and faster development techniques

Greater participation of 'citizen developers' (business users who have insignificant technical experience but are able to build business applications without involving technical teams) facilitates quick development of solutions more aligned to business needs

Automated configuration and monitoring through infrastructure-as-code reduces downtime and increases overall productivity and security

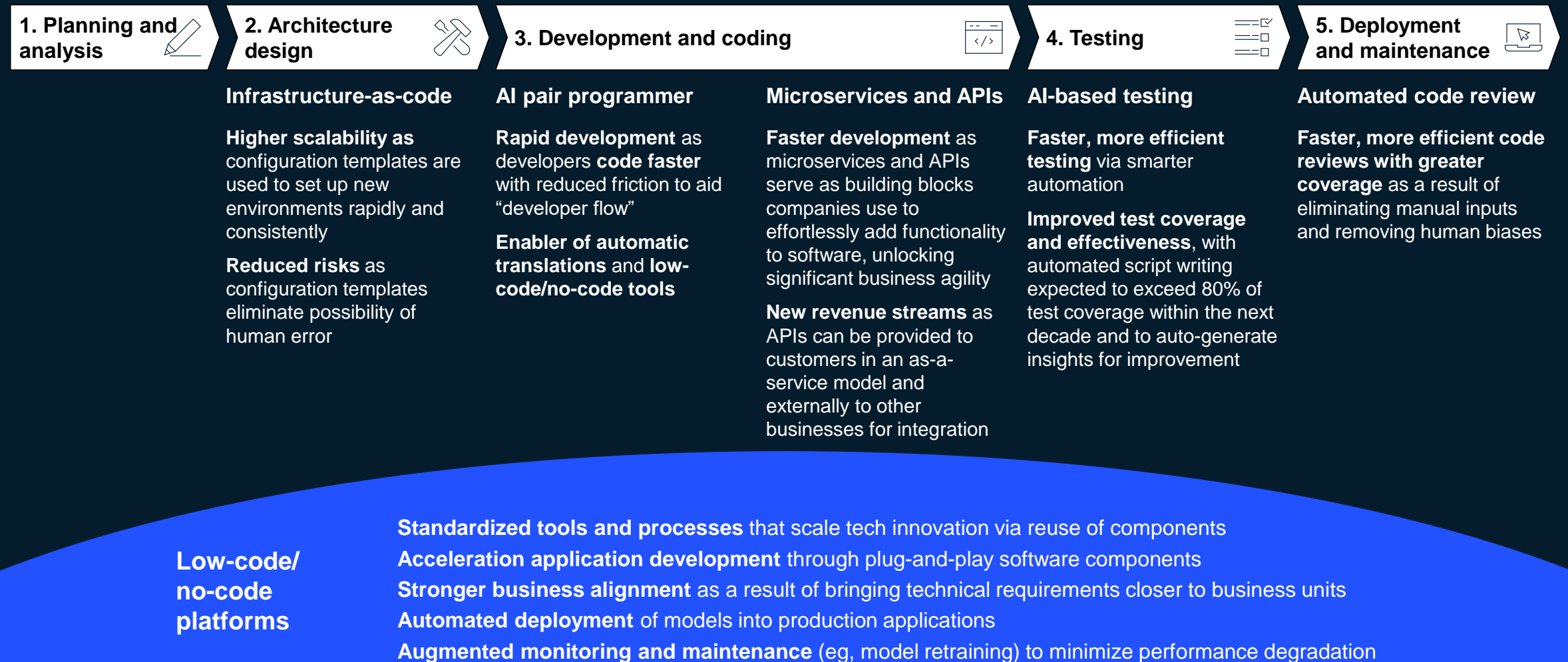
AI-based pair programmers are making solo developers more efficient and improving quality of code

Fully automated CI/CD pipelines enable lower disruption, higher code quality, and drastically shorter development cycles

What are the most noteworthy technologies?

Across the entire software development life cycle, technologies are already improving developer velocity

Not exhaustive







High industry relevance Medium industry relevance

What industries could be most affected?

Beyond the information technology and electronics industry, these technologies will have an impact on software development across all industries by reducing digitization challenges





Many industries are already reaping the benefits of low-code/no-code platforms, given their common qualities and requirements

Source: Expert interviews; McKinsey analysis

Industry	Examples	Common industry qualities	
Financial services 	Evolving business rules for processes such as onboarding, know your customer (KYC), and customer due diligence can be continuously handled by business analysts for efficiency	Compliance requires a wide variety of frameworks, protocols, and regulations , which typically vary by region, license agreement, etc	Heavily process-based industries Significant customization requirements Rapid pace of innovation to meet evolving customer needs
Healthcare systems and services; pharmaceuticals and medical products 	Case management processes for handling customer data, tailored and specific processes for high-risk patients, development and testing of new drugs, etc, can be customized by healthcare providers		
Manufacturing processes in automotive and assembly and aerospace and defense 	Production floor management allows industrial engineers to optimize operations, reduce training expenses for new developers, reduce production floor failures, and standardize safety/handover protocols		
Retail 	Consumer-friendly front-end applications can be rapidly created and tailored to the needs of an organization and its customers		

Who has successfully created impact with next-generation software development?

Leading players across industries have already leveraged advanced DevOps tools to optimize their SDLC¹

Stage of SDLC ¹	Technology	Example
 Architecture design	Infrastructure-as-code	Decathlon used infrastructure-as-code to automate infrastructure deployment , reducing deployment time from weeks to 30 minutes, allowing IT teams to focus on more complex tasks
 Development and coding	Automated CI/CD	Capital One leverages microservices and automated CI/CD to increase delivery speed without compromising quality through reusable building blocks and generation of templated pipelines
 Testing	AI-based test automation	Goldman Sachs uses the AI-based tool Diffblue Cover to generate unit tests for legacy software, leading to a 180x increase in the speed of writing tests for a core back-end application
 Deployment and maintenance	AI-based code reviews	Atlassian uses AI-based tools by Amazon Web Services to improve code performance by identifying code paths that demonstrate poor CPU ² utilization or latency

¹Software development life cycle.

²Central processing unit.

What uncertainties must be resolved for the trend to achieve scale?



Low-code/no-code platforms

Modest amount of customization is possible, compared with traditional programming languages

Monitoring and debugging applications is **difficult**, especially when they are integrated across several low-code/no-code platforms



Infrastructure-as-code

Comprehensive monitoring and version control is required to ensure errors do not spread across servers

Fragmented vendors could disrupt integrated applications, given uncoordinated changes and upgrades



AI "pair programmer"

Generated code **may be unusable or inefficient** and may have **security vulnerabilities**

Coders can be steered in the wrong direction if tools are not regularly updated with the standards or trained on clean, fast code



Microservices and APIs

Customizing APIs is difficult without significant time and effort

APIs **introduce security risks** by adding another attack layer that can be exploited



AI-based testing

Autonomous tools are typically specialized (eg, by programming language, test type)

Companies over-rely on automated testing/reviews when this tech scales; humans do not consistently check for errors in test and review outcomes



Automated code review

Tools **do not identify all defects and inefficiencies** in code

What are some topics of debate related to the trend?



Source: Expert interviews; McKinsey analysis

1 To what extent can no-code tech reduce the need for traditional software developers?

While low-code/no-code platforms help teams rapidly prototype or enable citizen developers to take over some of the work developers do, they are still not flexible enough to reduce development work at every stage of the software development life cycle (eg, when legacy systems require upgrades)

2 From a cultural standpoint, will teams—both developers and non-developers—embrace or resist next-generation technologies?

Automation technologies reduce time spent on development, which raises concerns for employees whose workflows are highly automatable; developers, testers, and analysts may be reluctant or eager to switch to new technologies, depending on job security, technical comfort, etc

3 What intellectual-property issues might affect code written by an AI application?

As companies leverage AI generation tools, there is a concern around ownership: Will the company that developed the application own it, or will it belong to the AI-enabled code generation tool provider?

4 To what extent will business units take responsibility for the ‘health’ of applications?

As next-generation software brings development capabilities to “citizen developers” embedded in business units, questions about organizational structure and responsibilities emerge—eg, as business users create applications, who is responsible for maintaining them?

Additional resources

Related reading

[Developer Velocity: How software excellence fuels business performance](#)

[Security as code: The best \(and maybe only\) path to securing cloud applications and systems](#)

[Developer Velocity at work: Key lessons from industry digital leaders](#)