diffblue
AI for Code

The Diffblue

Java Application Modernization Survey

How and why are enterprises modernizing critical Java applications, and what's holding them back?

Java remains king, but app modernization is held back by code quality

Organizations will modernize over 80% of their Java applications, and unit testing is playing a vital role in overcoming the key barrier to success: maintaining code quality

The Java App Modernization Survey

Java has been used to build enterprise software for more than two decades. A staggering amount of code has been written in that time and much of it is still in use today, supported by a development community numbering in the millions. How enterprises align their Java applications with modern engineering methods and new business strategies will be a critical question as they seek to gain competitive advantage through greater speed, agility and efficiency.

Diffblue's Java Modernization Survey sought to uncover more detail about what application modernization really means to the teams responsible for Java-based systems. We surveyed 450 Java application owners and engineering leads in the U.S. and the United Kingdom (UK) about the factors driving their modernization projects, the biggest challenges they expect to face, and what impact the process will have on development teams.

The results showed that enterprises face a daunting task. **Organizations expect to modernize over 80% of their existing Java applications** – which typically number in the hundreds for larger firms, and in the thousands for the biggest enterprises - and believe those modernization efforts will, on average, **divert 45% of available developer resources, potentially for 5 years or more**.

80%
of Java apps will be modernized

45%
of developer time could be needed

To better understand the implications of this massive undertaking we looked more closely at how engineering leaders described the drivers, enablers and challenges of modernization. 3 key insights stood out:

Security

Engineering leaders believe existing Java applications are not secure enough to support a modern business and its customers. **56% of respondents believe that 'improving application security' is a key driver of modernization in their organization** – 13% ahead of the next most common choice. 19% stated security improvements as the single biggest reason for embarking on modernization in the first place.

KEY INSIGHT 1

Code Quality

Code quality is the biggest barrier to successful application modernization. **'Maintaining code quality' topped the list of all major challenges (46%)**, and was highlighted as the single biggest blocker (by 18%). It is therefore unsurprising that 97% of respondents believe unit testing plays a 'very' or 'extremely' important role, despite less than 20% having the coverage levels they'd prefer.

KEY INSIGHT 2

Automation

Modernization is synonymous with automation. Many see it as a goal in and of itself: **58% of respondents said modernization means 'automating processes to accelerate change'** in their organization - the most popular choice. Regardless of the desired outcome, automation is seen as an important enabler of success: **90% of respondents expect to increase their use of automation** for Java development because of modernization projects.

KEY INSIGHT 3

In addition to these highlights, the ongoing importance of Java applications also came through loud and clear in the survey results:

87% believe Java modernization is a higher priority than the other projects they have planned

4% plan to migrate to other languages as part of the process.

As we delve into more detail in this report we'll use 4 main themes - the continuing importance of Java; drivers of modernization; the role of automation; and barriers to success - to explore and expand on these key insights.

1. The continuing importance of Java

> For many organizations, legacy systems are seen as holding back the business initiatives and business processes that rely on them. When a tipping point is reached, application leaders must look to application modernization to help remove the obstacles.
>
> **Gartner**

- The importance of Java shows no signs of diminishing
- Java modernization is a top priority for almost all IT teams
- Spring will remain central to Java applications

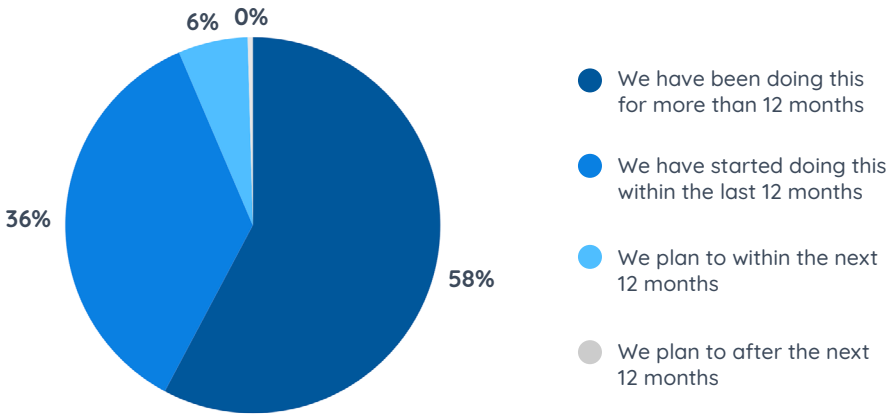
Despite the rise of alternative programming languages, it is apparent Java will continue to loom large in many firms for years to come. **96% of survey respondents consider Java applications to be either extremely or very important to their organizations**, so it's little wonder aligning them with a modern IT strategy is so vital.

The survey confirmed IT leaders are successfully getting the message across. In most firms Java modernization is already a funded project, not an aspirational goal, showing that many business leaders also understand the need for change.

59% of organizations have created a dedicated modernization budget to support their efforts; business project budgets are providing separate funds for 48% of respondents, while 49% said general operating budgets have increased.

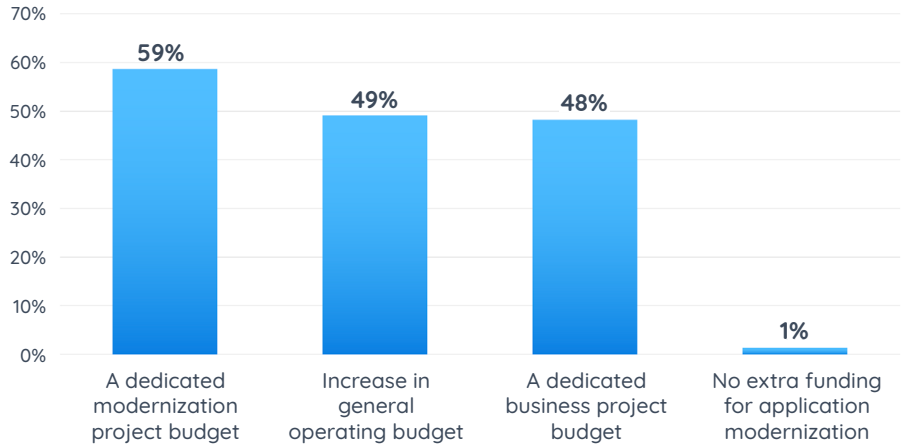
The relevance of the systems in question is further illustrated by the amount of ongoing development outside of modernization. On average, **65% of Java applications are still being developed**. This work – delivering value beyond bug and defect fixes – may be impacted by the diversion of resources towards modernization.

Java Modernization Plans



Q: Is your organization currently modernizing any of its Java applications?

Application Modernization Funding



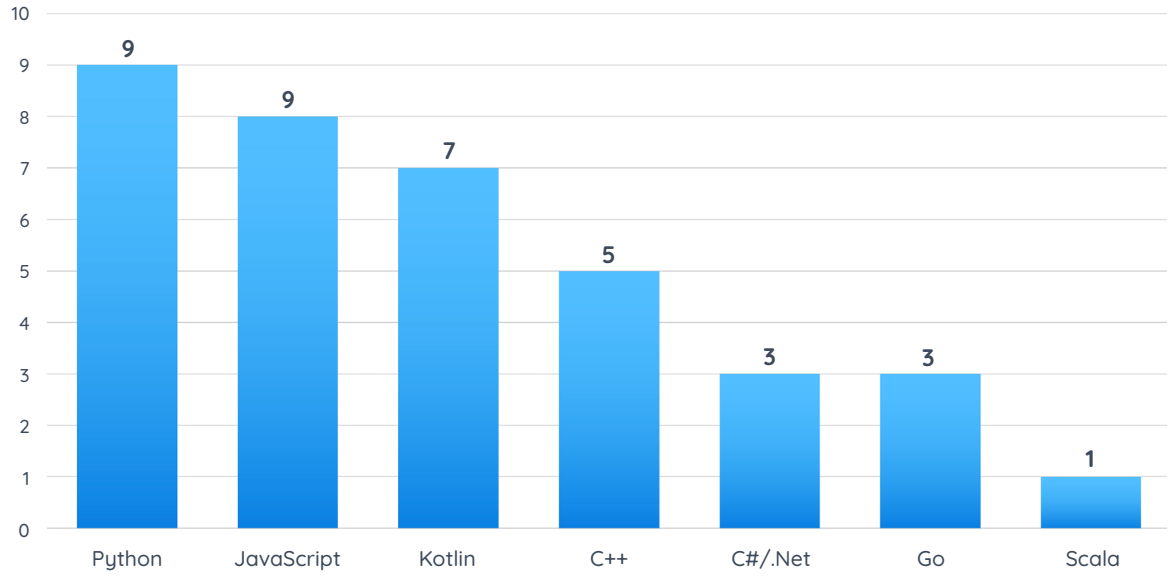
Q: Where does funding, if any, for application modernization within your organization come from?

Java modernization will be extensive, and funds are available to make it happen. But how does it stack up against the many other crucial projects firms may have in mind? **87% of Java application owners and technical leaders believe modernizing Java applications is a higher priority than other projects**. Application owners view modernization as particularly important, with 35% ranking it as 'significantly higher priority' (vs just 25% of engineering leads).

95% of respondents said the modernized versions of Java systems will also be written in Java, rather than migrating to another programming language.

The reality may be more nuanced however: some firms will take a 'hybrid' approach to modernized applications, with core codebases remaining in Java while new functionality is written in other languages like Kotlin.

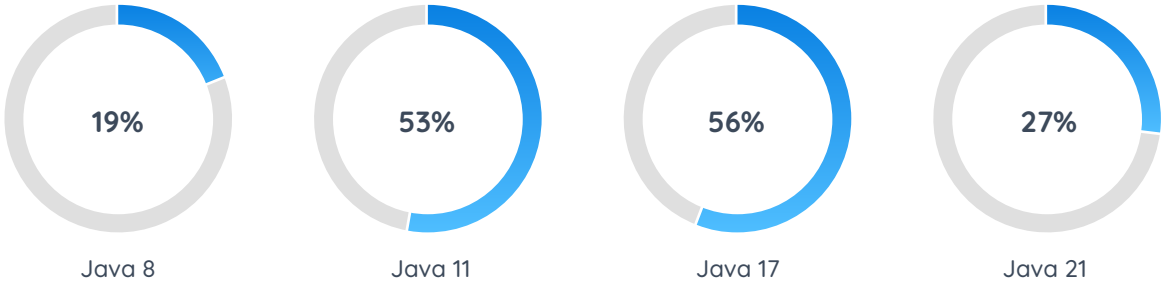
Only 5% of firms will migrate apps away from Java



Q: If any Java applications are likely to be re-written in different languages, which are most likely to be used?

87% of Java application owners and technical leaders believe modernizing Java applications is a higher priority than other projects.

Java versions in modernized apps



Q: If any Java applications are likely to be re-written in different languages, which are most likely to be used?

100% of respondents expect to use Spring in at least some modernized applications, with 54% expecting to upgrade to a newer version.

Many modernization projects will encompass an upgrade to newer versions of Java to take advantage of new features. Over half of the **respondents expect to have Java 11 (53%) and Java 17 (56%)** applications in their modernized landscape. Perhaps more surprisingly, a significant minority expect to keep using Java 8 for some modernized systems (19%) while some see Java 21 – not available when the survey was carried out - as part of the solution (27%). The latter outcome may reflect the anticipated duration of modernization efforts.

When it comes to frameworks, Spring continues to be the default choice for Java apps, despite a range of alternative options. **Every respondent expects to use Spring in at least some modernized applications**, with 54% expecting to upgrade to a newer version. Just 9% of people said they would even try another framework as part of their modernization plan.

2. What's driving modernization?

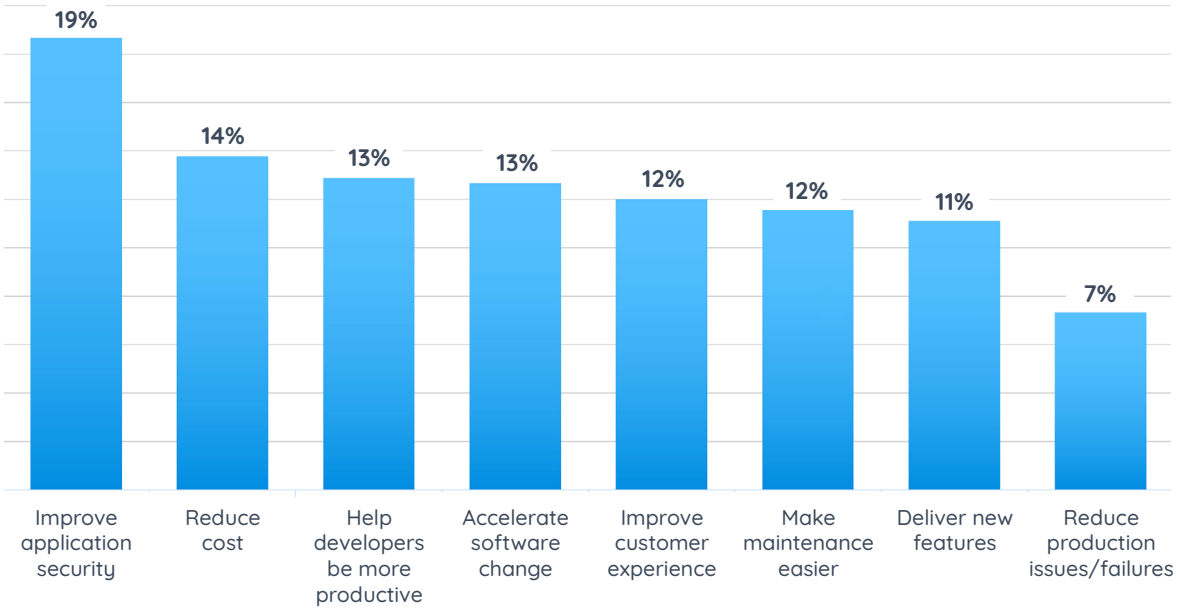
> More than anything else Java teams associate modernization with the increase in speed and agility enabled by automation: **58% said modernization means 'automating processes to accelerate change'** in their organization.

- Teams believe security is an important reason to modernize
- Modernization means automating to accelerate change
- The human cost of not modernizing is real and recognized

When we asked why firms were planning to invest so much time and effort in modernization, the survey results were emphatic, if perhaps surprising: **19% said they think 'improving application security' is the single biggest reason**. When they ranked their top three reasons for launching a Java application modernization initiative, **security is even further ahead of other factors (56%)**, beating customer experience (43%) into second place and developer productivity into third (39%).

Answers to this question also varied somewhat by job role. Perhaps unsurprisingly, for example, far more executives see increasing developer productivity as a goal (16% vs 10%), while engineering leads are more concerned with reducing failures and issues in production (10% vs 4%). Both groups chose application security as the most important reason, however.

Why modernize?

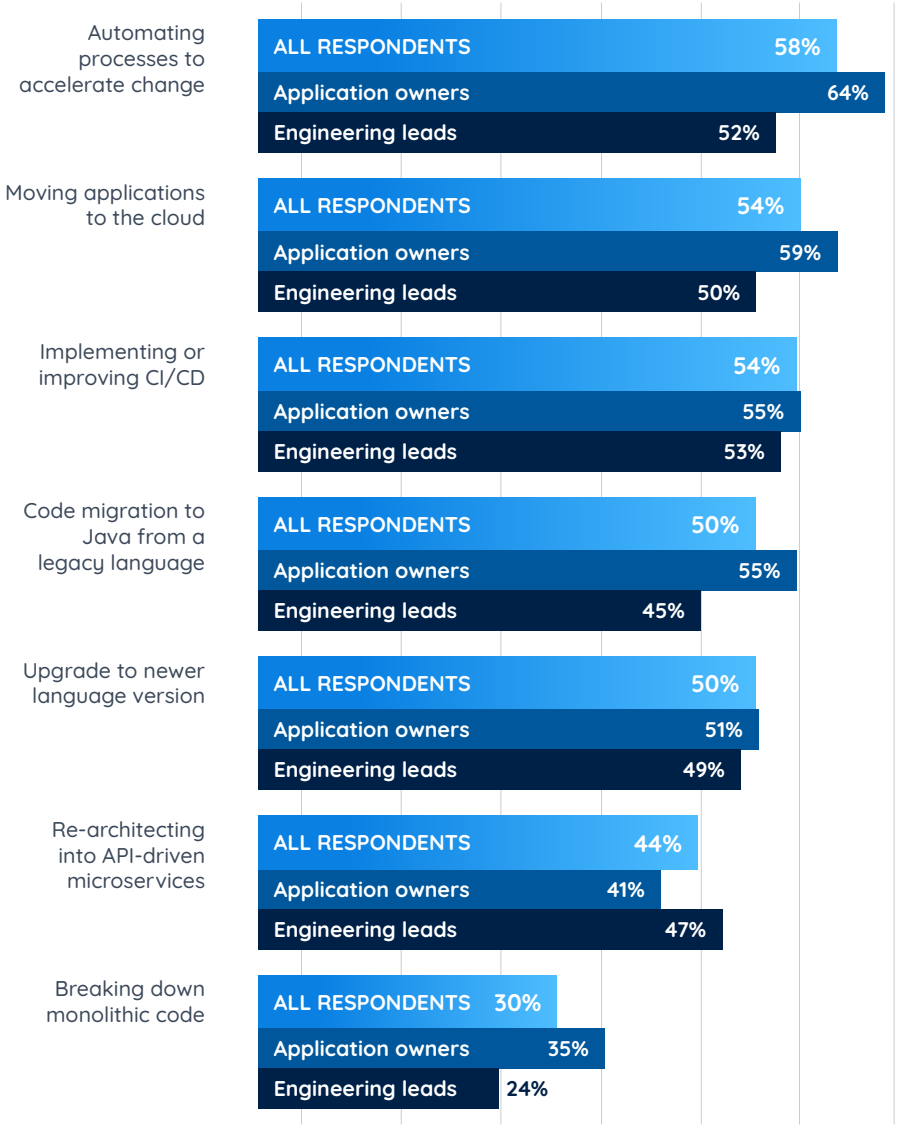


Q: What's the most important reason for your organization's modernization of its Java apps?

Modernizing to ensure security of mission-critical applications is especially important to larger firms.

31% of organizations with 5,000 or more employees ranked security as the most important reason for Java modernization, versus 19% overall.

What does modernization mean?



Q: What do you think application modernization means in your organization?

But 'modernization' is not a precise term, so what does it mean to IT leaders in practical terms?

A combination of cloud migration and moving to a microservices architecture is often seen – and promoted by many - as the de facto answer to this question. The survey respondents indicated things may be less clear-cut: only **54% said Java modernization means 'moving applications to the cloud' and 44% chose 're-architecting into API-driven microservices' as a goal.** Perhaps more unexpected was that **less than 25% believe they will do both**, presumably indicating that a significant amount of 'lift and shift' is planned. While the survey doesn't examine the specific reasons for this, considerations like security, operational continuity, and developer availability, as highlighted earlier in this report, presumably play a part.

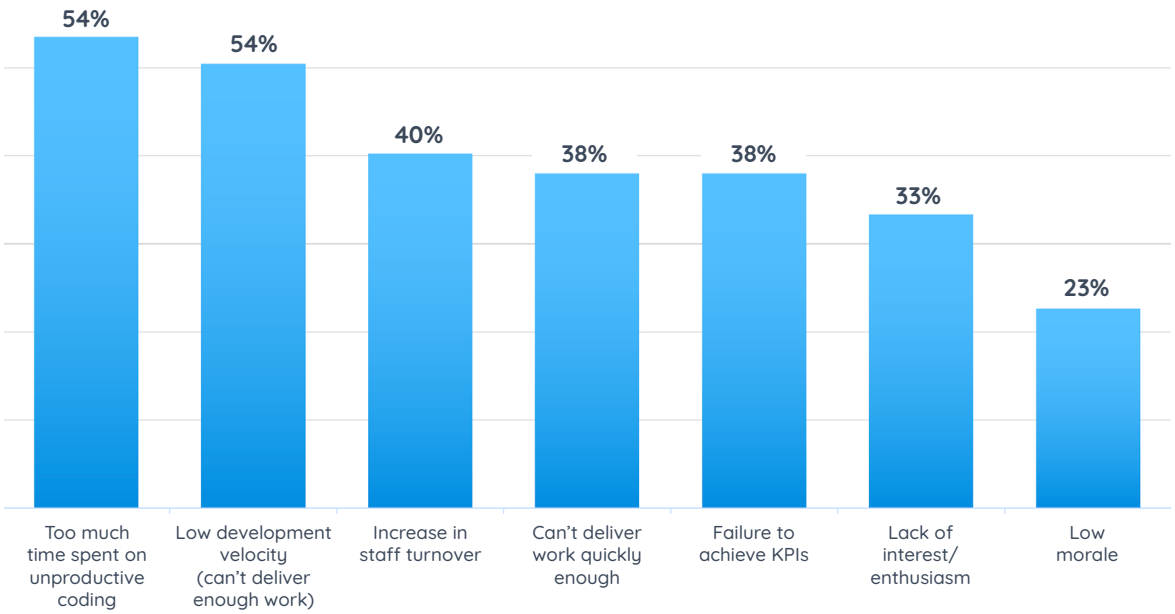
In fact, more than anything else Java teams say they associate modernization with the increase in speed and agility enabled by automation: **58% said modernization means 'automating processes to accelerate change'** in their organization (the most popular choice overall). In almost every case this focus on process automation was selected alongside other interpretations, underlining the importance of automation to modernization success regardless of wider context. Perhaps unsurprisingly, **'implementing or improving CI/CD' came in joint second (54%)**.

Application owners and engineering leads interpreted the meaning of modernization somewhat differently. Application owners seem to have a broader view of the term and were the most focused on automation, but it's also a key concern for engineering leads.

The converse of the reasons to modernize – weak security, poor customer experience, lower productivity, and so on – can be seen as key business risks of not doing so. But the survey also looked specifically at the impact on development teams of failing to modernize.

Though speed and productivity stood out here ('too much time spent on unproductive coding' at 54%, and 'low development velocity' at 50%), there was also a notable recognition of the **'human cost' of failing to modernize. 40% of respondents said it would result in an increase in staff turnover**; smaller but still significant cohorts **highlighted a lack of interest (33%) and low morale (23%) as possible outcomes**. It seems unlikely that improving developer experience would be seen as sufficient justification for modernization in most firms, but recruitment and retention are real challenges for many IT leaders today. As such, any positive impact in these areas is no doubt a welcome benefit of the application modernization process.

Consequences on development teams of not modernizing apps



Q: What do you think are the consequences for development teams if an organization's Java applications aren't modernized?

40% think failing to modernize Java applications will increase staff turnover.

3. Barriers to successful modernization

> There's another reason why application modernization should be a continuous process — you are never finished. Applications and software engineering leaders often conceptualize legacy applications as those old mainframe applications from the 1970s and 1980s. But when we are done modernizing those applications, new legacy applications will emerge — such as client/server applications from the 90s, and early Java and .NET applications from the 2000s. Technology ages quickly and requirements change constantly, which creates new legacy applications over time.

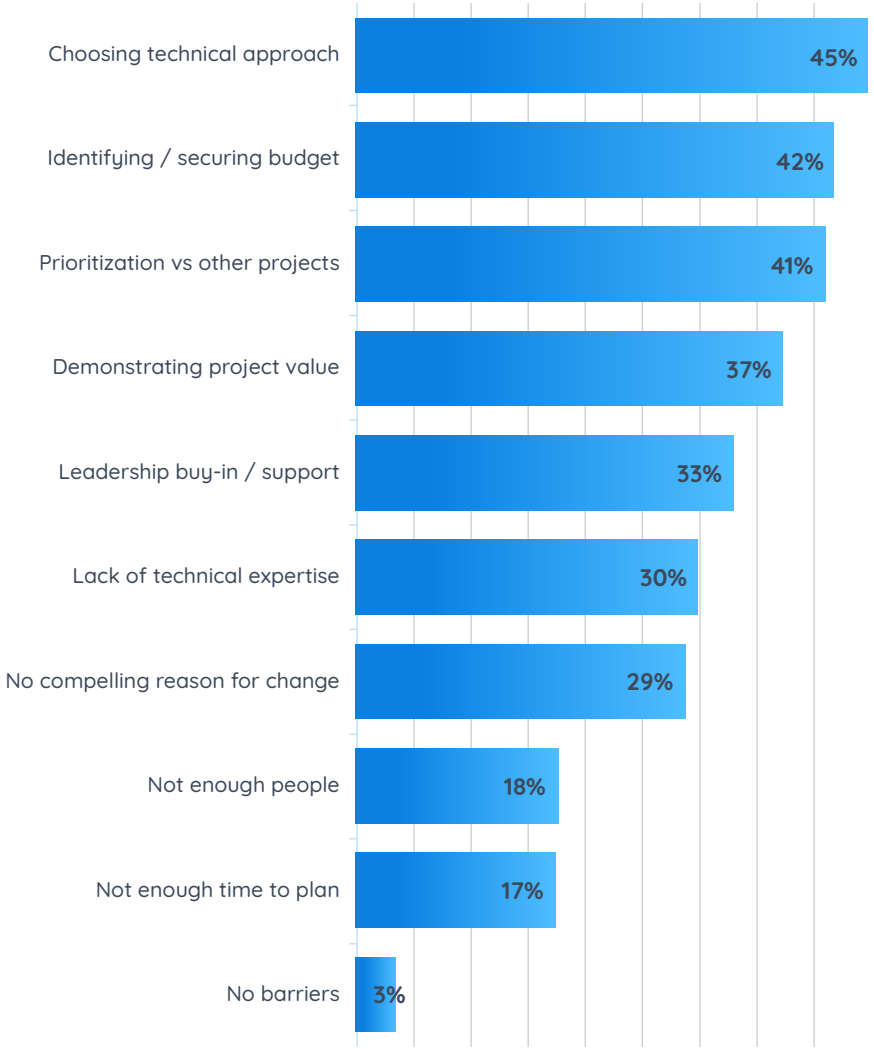
Gartner: Use Continuous Modernization to Optimize Legacy Applications, by Stefan Van Der Zijden, Deacon D.K Wan, Howard Dodd, July 20 2022

- Maintaining code quality is the biggest Java modernization challenge
- Teams expect the modernization resource requirements to be huge
- Unit testing is vital, but will be extremely labor-intensive

Though almost all Java teams plan to modernize their applications, challenges are abound. **The first is simply getting started. The survey found selecting the right technology or platform to be the most common roadblock (chosen by 45%)**, perhaps reflecting a struggle to define a clear strategic direction in today's dynamic and rapidly changing technology landscape. Securing budget (42%) remains a problem for a minority despite the general commitment to modernization; prioritization against other projects (41%) and demonstrating value (37%) are all also seen as significant hurdles.

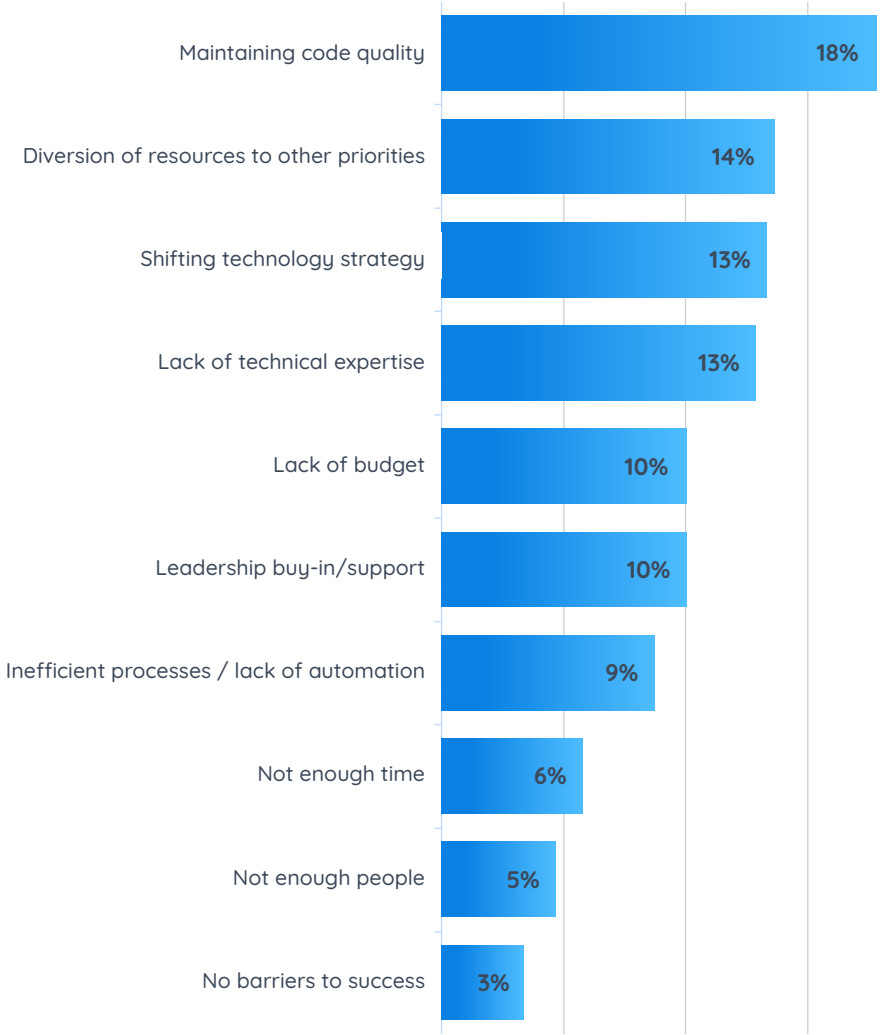
Almost no-one (3%) believes nothing stands in the way of modernization projects, despite the general recognition of their importance noted earlier in this report.

Biggest roadblocks to getting started



Q: What do you think are the biggest barriers to starting application modernization projects in your organization?

Biggest barriers to successful modernization



Q: What do you think are the biggest barriers to application modernization success in your organization?

Unsurprisingly, the survey also finds that respondents don't expect modernization to be plain sailing once they're up and running. **Survey respondents identified 'maintaining code quality' as the stand-out barrier to success**. It was most commonly cited (46%) of all expected challenges, with 18% selecting it as the single most important. This is likely a reflection of the need to maintain operational continuity of important applications during and after modernization, and difficulty in doing so when working on large, complex, often poorly documented codebases: just **22% of respondents said most of their Java apps were built by in-house developers**.

Shifting technology strategy/platforms (43%), diversion of resources (41%), a lack of technical expertise (34%) and inefficient processes/ insufficient automation (32%) round out the top 5 challenges.

Engineering leads see maintaining code quality as more of a challenge than application owners.

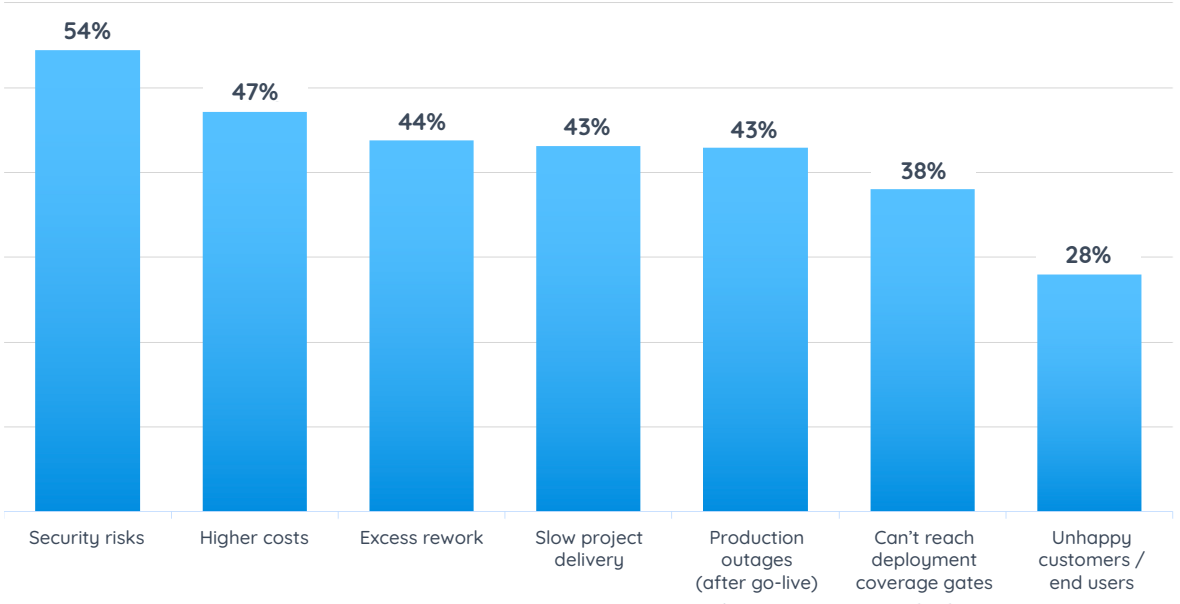
51% put it in their top 3, compared with 41% of execs.

Unit testing is recognized as a powerful way to address this code quality challenge. **97% of survey respondents said unit testing is extremely or very important** when modernizing operational software. (Unit tests provide fast feedback to developers and excel at identifying regressions when changes are made to existing code – two critical goals during application modernization.)

Java teams foresee a wide range of negative consequences if insufficient unit testing is carried out. **Security topped the list once again, with 54% identifying it as a concern**, above higher costs (47%), excess rework (44%) and production outages after go-live (43%).

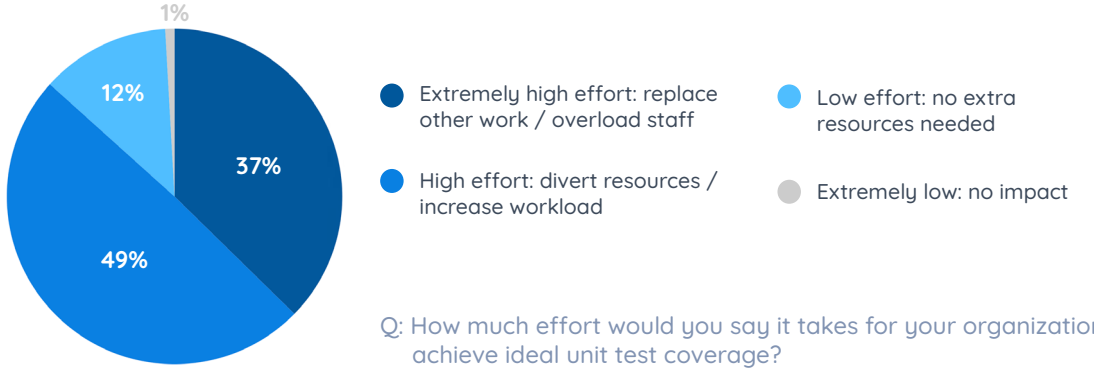
On average, **77% was seen as the ideal level of unit test coverage** – a figure achieved by **less than 20% of respondents** today. A large majority (87%) believe a high or extremely high amount of developer effort would be required to achieve the ideal coverage level, despite already claiming an average of 57%. In fact, **30% think an extreme amount of effort would only get them a 'sufficient' amount**, despite their existing tests. It seems reasonable to assume that unit testing is a key contributor to the huge amount of developer effort – **45% of available developer time, on average** - firms expect to need for modernization. No wonder 'diversion of resources' is considered a serious risk factor.

Consequences of insufficient unit testing



Q: What are the most common consequences of insufficient unit testing during application modernization?

Effort needed to achieve ideal unit testing levels



Q: How much effort would you say it takes for your organization to achieve ideal unit test coverage?

Even if enough developers are available to support creation and maintenance of the unit tests needed for successful modernization, it's work that most don't want to do. A previous Diffblue survey found that **82% of Java developers would prefer not to work on repetitive tasks like unit test writing**, and almost 40% wished they didn't have to write them at all.

Despite the challenges, the pace of Java modernization is accelerating. Nearly two-thirds of organizations **(65%) plan to update between 100 and 1,000 Java applications**. Though most teams started work in 2022, survey respondents believe they will only modernize around a third (35%) in 2023. By 2025, that percentage is expected to rise to 58%.

Even with this increased focus, **a typical organization will not have completed their goal of modernizing over 80% over Java applications within the next 5 years** (by 2027), perhaps because early projects represent the lowest hanging fruit with the greatest chance of success. By this time new modernization requirements may have surfaced, such as migration to a newer version of Java. In fact, as technologies continue to evolve, application modernization may end up not so much a task to be completed as a permanent state of IT affairs.

Larger firms see a lack of technical expertise as more of a problem when it comes to application modernization.

44% of organizations with 5,000 or more employees ranked it as one of their top challenges, vs 34% overall, perhaps reflecting the greater difficulty inherent in understanding how legacy code works in large applications.

THE SCALE OF THE CHALLENGE

Most organizations with more than 3000 employees have at least 100 Java applications.

3000 EMPLOYEES

100+ JAVA APPLICATIONS

96% of companies have at least 10 Java applications.

96% OF ALL COMPANIES

10+ JAVA APPLICATIONS

4. The importance of automation in modernization

> **Automation is clearly intrinsic to success:** a full 90% of survey respondents stated they expect application modernization to drive an increase in their use of automation for software development.

- **Automation is an intrinsic part of successful application modernization**
- **There's no single fix: a variety of tools will be needed**
- **CI pipelines will play a key role, but teams must align on the investment required**

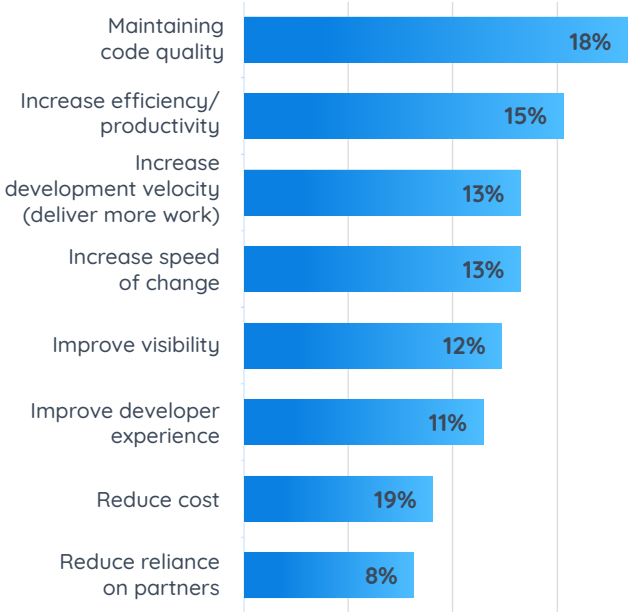
Modernization of Java applications is occurring alongside an ongoing shift left of responsibility toward developers. Automation is commonly seen as a critical enabler of both these initiatives as organizations look to deliver high quality code at a faster pace.

58% of Java teams chose 'automating processes to accelerate change' as a definition of application modernization in their organization. But even for those that didn't, **automation is clearly intrinsic to success: 90% of survey respondents stated they expect an increase in their use of automation** for software development as a result of application modernization efforts.

The survey finds a range of reasons for this increase, which unsurprisingly map closely to the reasons for modernization. **Improving security stands out once more, with 50% of respondents choosing it as a reason for more automation**; 18% said it was the single most important factor. Increasing productivity (45%), higher development velocity (42%), and **improving developer experience (40%)** were also ranked highly.

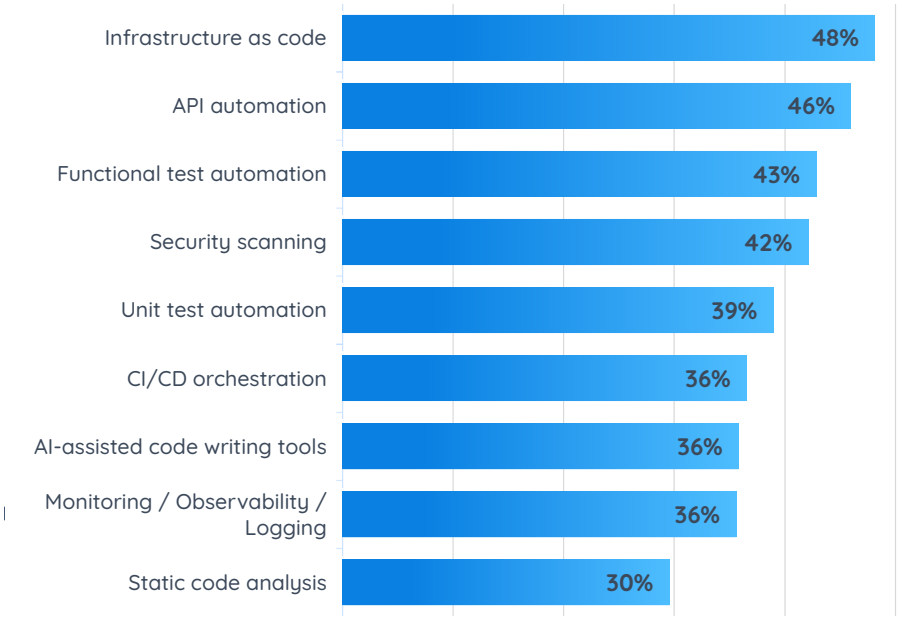
This last data point is perhaps surprising in the context of a major IT project, but further reinforces the staffing challenges companies face, and awareness among IT leaders of the importance of developer job satisfaction. The positive **impact of automation on developer experience is significant enough for it to be selected by 11% of respondents as the single most important reason for employing more automation** during modernization.

Why employ automation?



Q: What are the main reasons for employing automation as part of your organization's application modernization project?

What automation?



Q: What automation will your organization employ as part of its application modernization effort?

Large organizations see more need for automation that helps them move quickly.

51% of respondents from a firm with 5,000 or more employees said increasing development velocity is a key reason for investing in automation to support modernization, vs 42% overall.

Respondents expect to adopt a range of automation to support their modernization efforts – **over 80% will use at least 3 different types**. Common uses of automation include employing infrastructure as code (48%), API automation (46%), security scanning (42%) and unit testing (39%).

Continuous integration (CI) and DevOps appear to be seen in a similar light to automation by Java teams; that is, both as a reason to modernize and a key part of the process (perhaps no surprise given automation is also fundamental to CI). **54% selected 'implementing or improving CI/CD' as a basic meaning of application modernization** but **90% said they plan to use existing or enhanced pipelines** (just 8% will adopt CI and/or DevOps for the first time), perhaps implying that firms feel their application development and delivery processes are insufficiently mature today.

Once again, application owners and engineering leads have a slightly different view of this matter. More owners expect to use an existing pipeline (41% vs 32%) whereas more engineering leads feel enhancements will be required (57% vs 50%). This may simply be a result of day-to-day familiarity with existing tools, but sets up the potential for conflict: optimization of CI pipelines is not a trivial matter, especially across large organizations. Teams should attempt to align on the question of exactly what automation is required – for CI/CD and elsewhere – as early as possible in the modernization process.

Automation spotlight

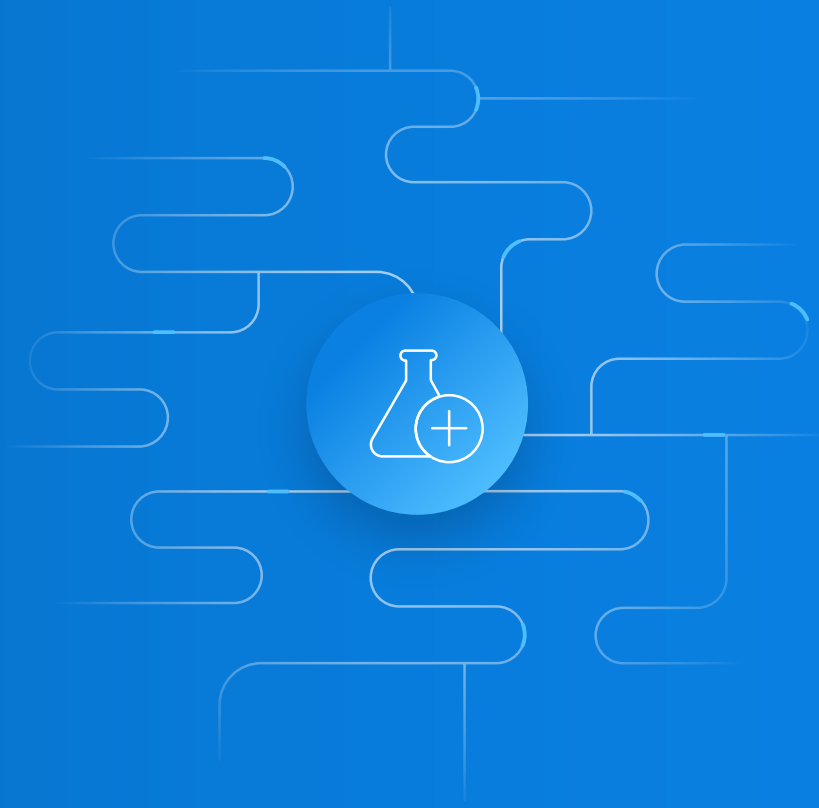
The role of AI in application modernization

AI-powered code-writing tools are providing ways to minimize tedious, repetitive, error-prone development work like writing boilerplate code, API calls and unit tests, so developers can move faster and focus on more productive work.

Diffblue Cover gives enterprises and Java developers the means to unlock more value from unit testing – rated by 97% of organizations as a very important element of application modernization. Its core technology uses reinforcement learning AI to autonomously write JUnit tests that are indistinguishable from those created by a developer.

Cover operates at high speed. It can write more tests in hours than a typical developer could create in a full year **– a huge benefit when modernizing old, complex legacy code that starts without good test coverage - and automatically updates your entire unit test library whenever subsequent code changes are made during your modernization project. For new or existing code, on the desktop or embedded in a CI pipeline, Cover eliminates tedious, repetitive, error-prone work from developer workloads.**

A range of additional features leverage this core AI for Code platform to further accelerate development and reduce risk, including Cover Reports, which visualizes the state of unit test coverage across all your Java modernization projects and helps you identify where risk really lies.



Discussion and takeaways

Software is becoming ever more important in the enterprise, and Java applications continue to drive many core business processes. Modernizing those applications in a way that makes them more scalable, responsive, resilient and accessible is critical as organizations not only look to become more efficient, but also remain relevant in an era where reputation is directly impacted by the quality of digital experience.

The Diffblue Java Modernization Survey appears to confirm that Java teams recognize this imperative and are acting on it.

Modern Java = Safer Java

The emphasis on security as an important part of Java modernization projects was one of the clearest outcomes of the survey. Improving application security was identified as the main driver for modernization by 19% of respondents, and was the most frequently selected reason overall (56%). Security was also ranked as the most important reason to increase use of automation, and one of the key reasons why comprehensive unit testing is considered so important.

Though clearly a concern for every business these days, this focus on security as a motive for modernization - ahead of oft-discussed 'transformation' benefits like speed, productivity and customer satisfaction – is notable. This result might simply reflect the age, complexity and criticality of much Java software – a combination of factors which may cause Java applications to be seen as a particular business risk.

Unit testing addresses the biggest challenge

It was perhaps also surprising to see 'maintaining code quality' ranked above concerns like expertise, people, money and time, which are often more fundamental barriers to the success of large IT projects. Those topics were still identified as challenges by a significant proportion of respondents, but perhaps a relatively higher prioritization of quality is a result of the commitment organizations have already made to Java application modernization. It also presumably reflects the fact that part of the challenge in modernization projects, unlike in novel development, may be ensuring things do not change (at least to the end user).

Unit testing is clearly recognized (by 97%) as an important part of the solution to the code quality challenge – no great surprise, given the amount of code refactoring likely to be involved in modernization - but getting the necessary coverage is far from easy. For a significant minority it's going to take an 'extreme' amount of effort to even get to 'sufficient'. 'Ideal' seems a long way off in such cases, especially if you're already diverting nearly half of development resources towards modernization.

Automation is key

It seems abundantly clear is that organizations believe application modernization can't succeed without automation. The task at hand is presumably just too big, complex and prone to risk to leave it entirely in the hands of human team members, especially when you consider that it might effectively be a never-ending process. In fact, most people see automation as one of the very meanings of modernization.

Unit testing is an area ripe for automation. On average, writing a single unit test can take a developer 10 minutes. Over the course of an application modernization project thousands of tests may need to be written – bringing us to the 'extreme' amount of effort required. Diffblue Cover helps to solve this problem by providing automation that frees developers to work on what you need them for most - writing meaningful code that delivers the modernization of business-critical applications.

Modernization impacts job satisfaction

Though data is limited, a less definitive but nonetheless interesting implication of the survey is that firms see legacy applications as hindering efforts to recruit and maintain the best development talent in today's highly competitive labor market. 40% of respondents said they believe a failure to modernize would result in an increase in staff turnover; smaller, but still significant, cohorts highlighted low morale (23%) and a lack of interest (33%) as possible outcomes. 11% also see better developer experience as the single most important reason to adopt more automation.

This further emphasizes the focus placed by many firms today on developer satisfaction and retention – a clear result of the amount of software that needs to be written and the insufficient amount of developers who are qualified to do so.

Curious how Diffblue can help?

If you'd like to learn more about how Diffblue's AI code-writing technology can help your enterprise to accelerate modernization by addressing the challenge of unit testing, visit diffblue.com/schedule-a-demo.

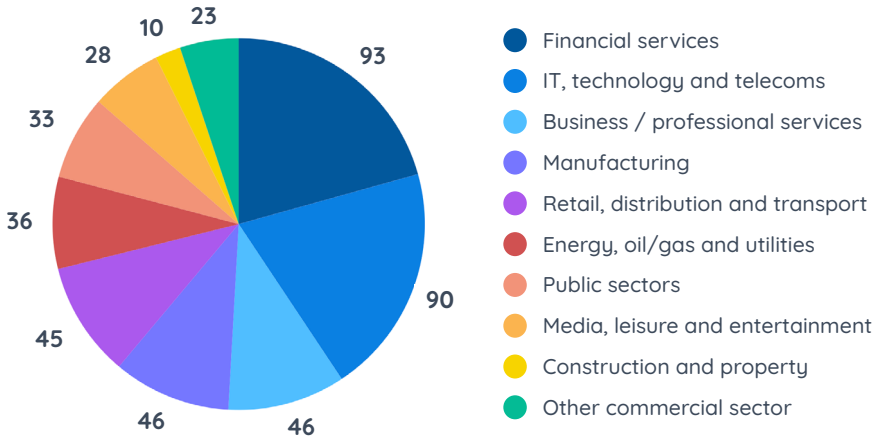
diffblue
AI for Code

Appendix: Survey Demographics

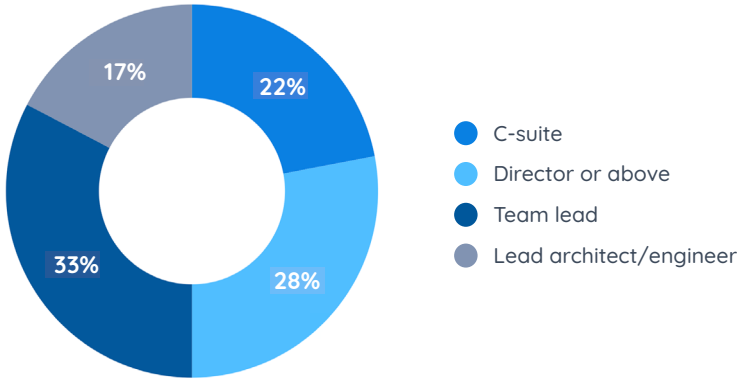
Diffblue worked with market research firm Vanson Bourne to distribute a survey to 450 respondents in the autumn of 2022. The questions included a mixture of Likert scales, multiple choice, and open-ended responses. Of the respondents, 300 were based in the US and 150 in the UK; all worked in application development at companies with at least 500 employees. Most work in organizations with at least 100 Java developers; some with more than 1000.

Industry sectors surveyed included financial services, IT/Technology, business and professional services, retail and various others. Participants all had job titles that fell in one of two seniority levels: Java application owners (VP, Head or Director level roles), and engineering leads/senior engineers.

Respondents by industry

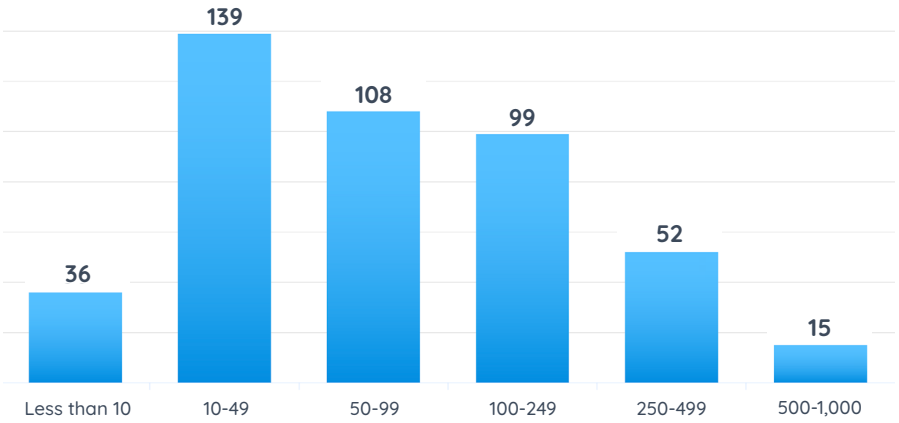


Respondents by job role

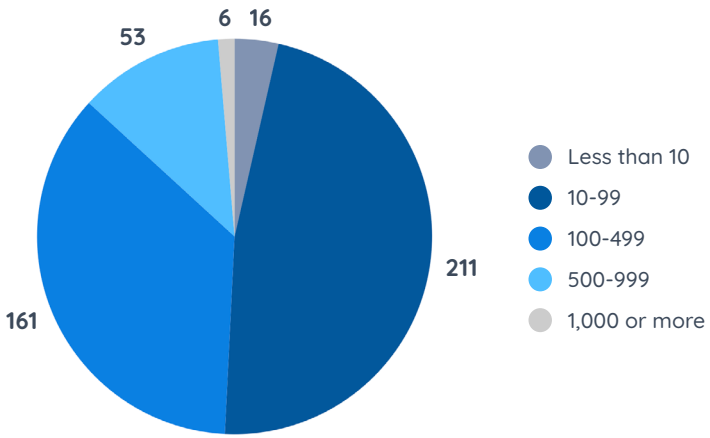


Appendix: Survey Demographics

Number of Java developers



Number of Java applications



About Us

Diffblue is a leading pioneer of software creation through the power of AI. Founded by researchers from the University of Oxford, Diffblue Cover uses AI for Code to solve the problem of effective unit testing. Capable of writing unit tests 250x faster than a human developer, Cover helps software teams improve code quality, expand test coverage and increase productivity, so they can ship Java software faster, more frequently, with fewer defects. Enterprises to have benefited from the power of Cover include Goldman Sachs, Citi and JP Morgan Chase.

To find out more visit diffblue.com