Report

2021 Spring Framework User Survey

How does using the Spring framework impact testing practices and code quality?

Key Findings

Since its release in 2003, the benefits of the **Spring Java framework** have made it the Java framework of choice for many organizations.

Last September, VMWare released the **2020 State of Spring report**, asking Spring users questions about what drew them to the framework and keeps them using it. We wanted to learn the specific benefits of the framework, and ask (for the first time, to our knowledge) how Spring has an impact on the testing practices and code quality of Spring users.

We found that there is a correlation between using Spring/Spring Boot and having better tested code. Specifically, Spring users:

- Highly value Spring: 86% of respondents use Spring/Spring Boot, and 96% benefited from using it
- Rank Testability fourth as a priority behind writing code that is reliable, modern and maintainable
- Spend 25% more time writing unit tests than non-Spring users (25% vs. 20%)

- Are more likely to say their organization's ability to test code is 'Excellent' (54%, compared to only 44% of Non-Spring Framework Users and 39% of No-Framework Users)
- Report higher code coverage (all of the respondents who reported 100% Java unit test coverage were Spring/Spring Boot users; a quarter of Spring users say they have 76%-100% coverage)
- Are the most likely to agree that unit tests make it easier to modernize legacy code (93% of Spring users agreed, compared to 88% of non-Spring users and 79% of no-framework users) and migrate to the cloud (94% of Spring users agreed, vs. 80% of non-spring framework users, and 74% of no-framework users)
- Report that DevOps and Cybersecurity essentially tie for top organizational priority (>90% for both) followed by Cloud Adoption at 47%

This report will go into greater detail on these findings and more.

Methodology and Demographics

Diffblue worked with Vanson Bourne to distribute a 15-question survey to 450 Java developers (300 in the US and 150 in the UK, at a mixture of small and large organizations). The questions included a mixture of Likert scales, multiple choice, and open-ended responses.

Of our respondents, 81% identified as male and 19% as female. The majority (53%) were between 35-44 years of age:

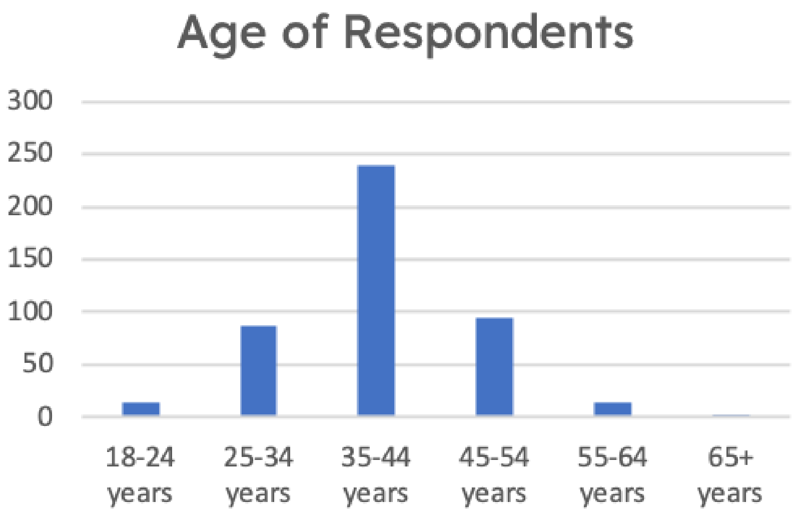

Figure 1: Breakdown of respondent ages

The intended respondents in this survey were Java developers who use Spring and other frameworks. It was estimated that approx. 60% of responses would come from Spring/Spring Boot, and 40% would be from people who don't use Spring. Interestingly, the majority of respondents to the survey are users of Spring and/or Spring Boot. Out of 450 respondents, 25 do not use Spring/Spring Boot but do use other frameworks, and 38 don't use any framework.

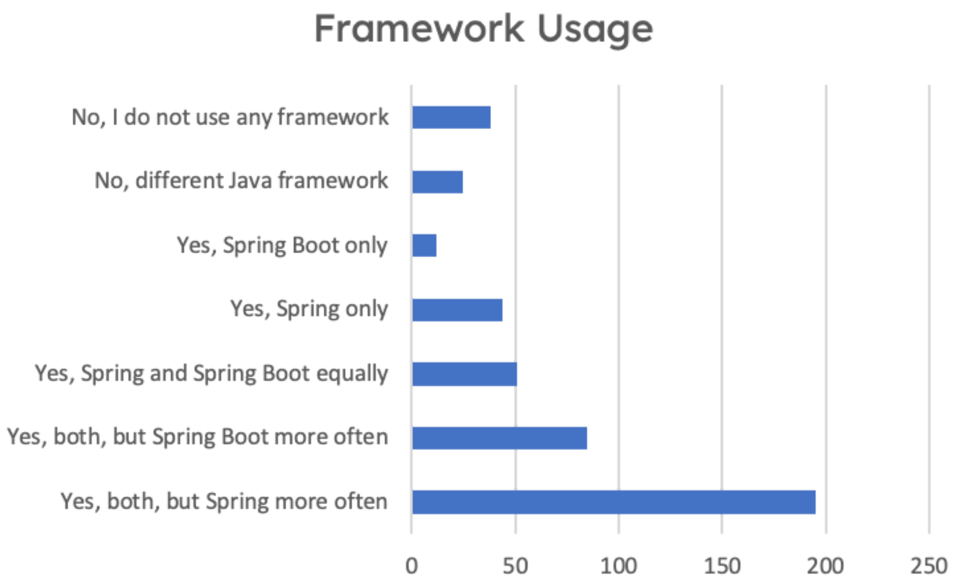

Figure 2: Reported usage of Spring, Spring Boot, other Java frameworks, or no Java frameworks

Sectors included IT/computer services, financial services, banking, insurance, and various others:

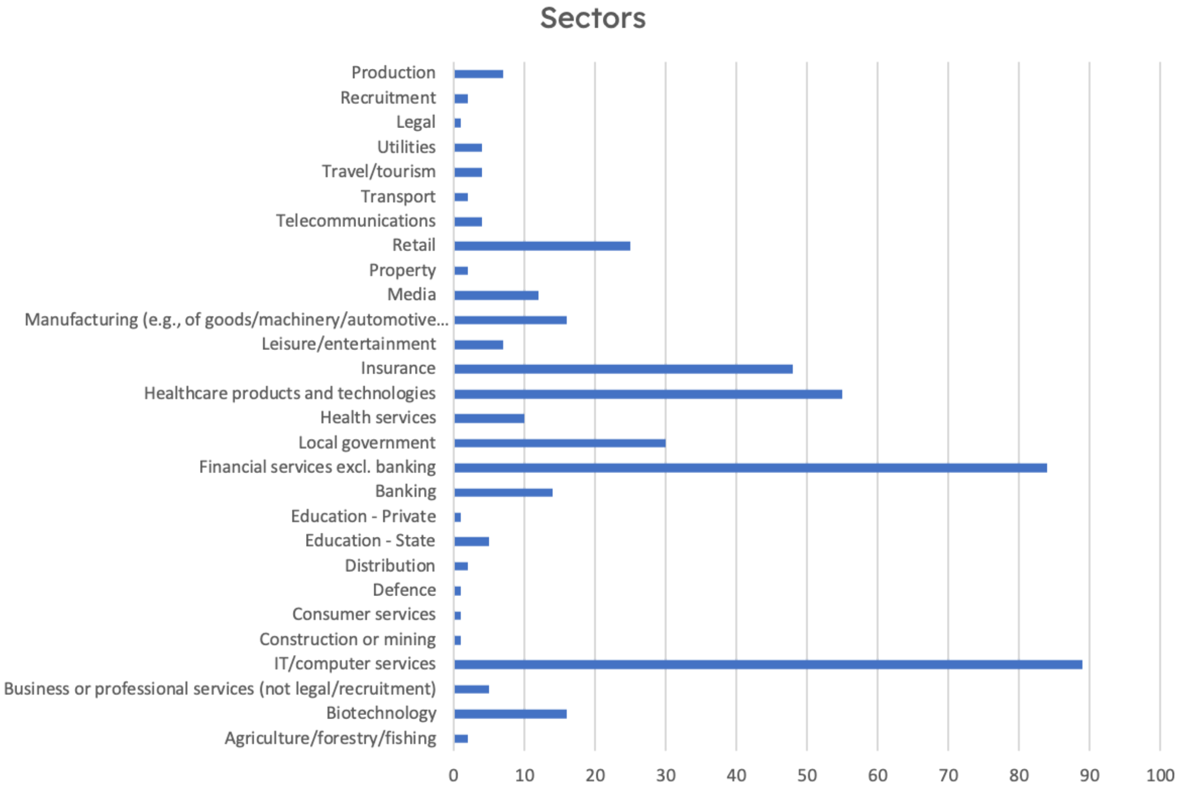

Figure 3: Sectors with at least one respondent

Participants all had job titles in technical positions, with most working in DevOps roles:

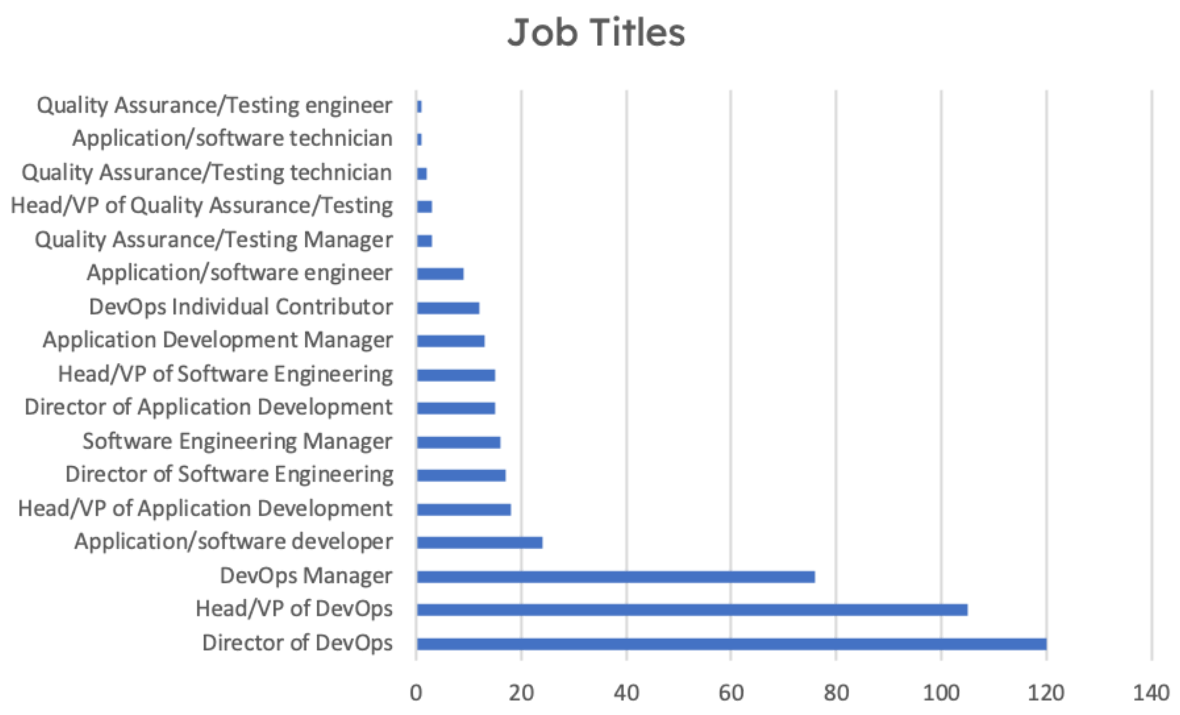

Figure 4: Breakdown of participants' job titles

Besides Java, the next most popular languages this group codes in are JavaScript (selected by 45.8%) and Python (37.3%).

Presentation of Findings

In Part One, the benefits of Spring according to users are laid out, as are answers to other general development preferences. In Part Two, we look at patterns in the differences in testing and code quality between Spring Framework users (SFU), users of other non-Spring frameworks (NSFU), and Java developers who don't use any framework (NFU). Finally, Part Three is a comparative analysis of other organizational priorities between framework users.

Part One: Java developers' general preferences

In this section, we will summarize the answers to questions about general preferences and opinions, including the benefits of Spring, the usefulness of a variety of testing tools, and development practices as a whole.

Most useful libraries

Respondents were given a free response option to share the single most useful library for their day-to-day working life. 48% said "I don't consider any library to be useful in my role," 6% chose "Don't know," and 46% submitted a response.

The most popular response was HTTP library (23 mentions out of 146), followed by PDF library (21), Excel reading library (9), Collection libraries (8), Messaging libraries (6), and a tie for Bytecode libraries and Scala library (5).

(Other honorable mentions include the Library of Congress and the Technology for Everyday Life courses at Watertown Library in Massachusetts).

Most useful testing tools

Respondents were asked to rate the usefulness of a variety of testing tools, from 'not at all useful' to 'very useful': JUnit, JMockit, Serenity, Mockito, PowerMock, Diffblue Cover[1], Parasoft Jtest, TestNG, Spock and Cucumber. Since the majority of these tools are fairly well known and popular, it was expected that the responses would lean more towards 'useful' than 'not useful', and this was confirmed in the findings.

However, the differences in usefulness were minor, but interesting: JUnit had the highest proportion of responses for 'very useful' at 44%, and another 40% said it was 'somewhat useful'. TestNG had 40% 'very useful' and 46% 'somewhat useful'. Diffblue Cover came in third for 'very useful' (39%) and 43% 'somewhat useful'. See the table below for the full results in descending order of 'very useful'-ness:

Tool Name	‘very useful’	‘somewhat useful’	‘not used at my organization’
JUnit	44%	40%	6%
TestNG	40%	46%	8%
Diffblue Cover[1]	39%	43%	10%
Parasoft JTest	38%	47%	9%
PowerMock	37%	46%	7%
JMockit	37%	44%	10%
Mockito	36%	45%	9%

Serenity	36%	44%	10%
Spock	33%	48%	11%
Cucumber	33%	41%	12%

[1] Diffblue Cover is owned by Diffblue Ltd. Participant data was collected anonymously via a third party and did not intentionally include any Diffblue Ltd customers.

Specific benefits of Spring or Spring Boot

5 options were provided as benefits of Spring/Spring Boot, plus 'Other' and 'Don't know':

1. Core technologies (e.g. Spring context, dependency injection, etc.)
2. Testing support
3. Data access
4. Integration with other technologies (e.g. Hibernate)
5. Easier setup of web interfaces/APIs

In the responses, all benefits were chosen fairly evenly, but 'Easier setup of web interfaces/APIs' was ranked first by 23.8%, and 'testing support' was ranked first by 21.2%.

Respondents were also asked to select all that applied for another five potential benefits: 58% of Spring or Spring Boot users reported the framework had saved time for developers, and 56% said they had experienced better supported unit testing as a result of using it. 54% agreed they had experienced testing acceleration, and 51% experienced smoother modernization. Finally, 49% said they had reduced developer friction as a result of using Spring or Spring Boot. (4% said their organizations experienced no benefits as a result of using Spring or Spring Boot):

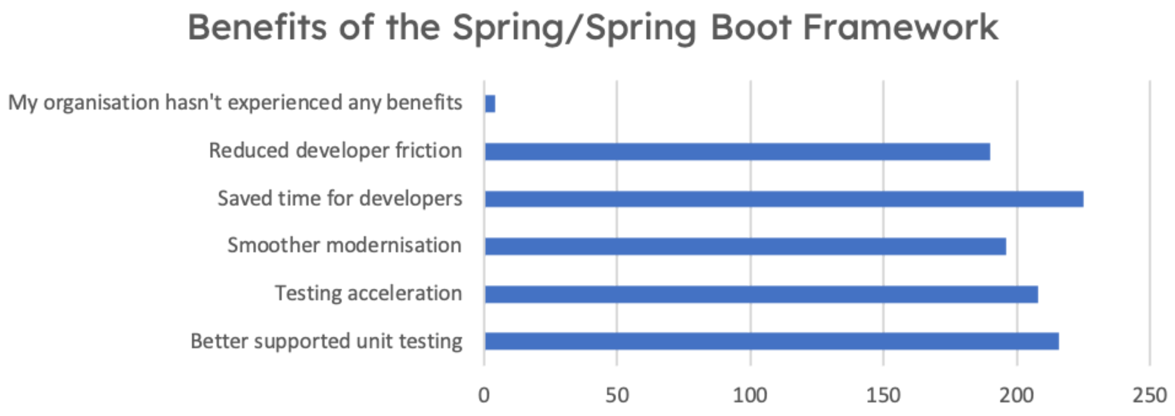

Figure 5: Benefits of the Spring and Spring Boot Framework

94% agreed that their organization had found it much easier to modernize as a result of using Spring or Spring Boot, and 6% said it was slightly harder.

Part Two: Comparative analysis of testing practices and code quality between framework users

In this section, we will look at the testing practices between Spring Framework users (SFU), Non-Spring Framework users (NSFU), and No-Framework users (NFU), as well as the potential impact of these practices on code quality.

Testing practices

Everyone we surveyed spends a lot of time on testing: SFU and NFU spend an average of 25% of their time writing unit tests, compared to 20% of NSFU. SFU also spend the most time writing other (non-unit) tests: 22.5% of their total time, compared to 21.8% of time for NFU and 19.3% of time for NSFU.

How does all of this time spent unit testing relate to the code coverage levels in respondents' organizations?

45% of respondents say their organization's Java code is less than 50% covered by unit tests. 37% say their organization's Java code coverage falls between 26-50%. 30% say it falls between 51-75%. 25% report a remarkable 76%-100% coverage (and 6% say they have a perfect 100% of code coverage). All of the respondents who said they have 100% code coverage are Spring users.

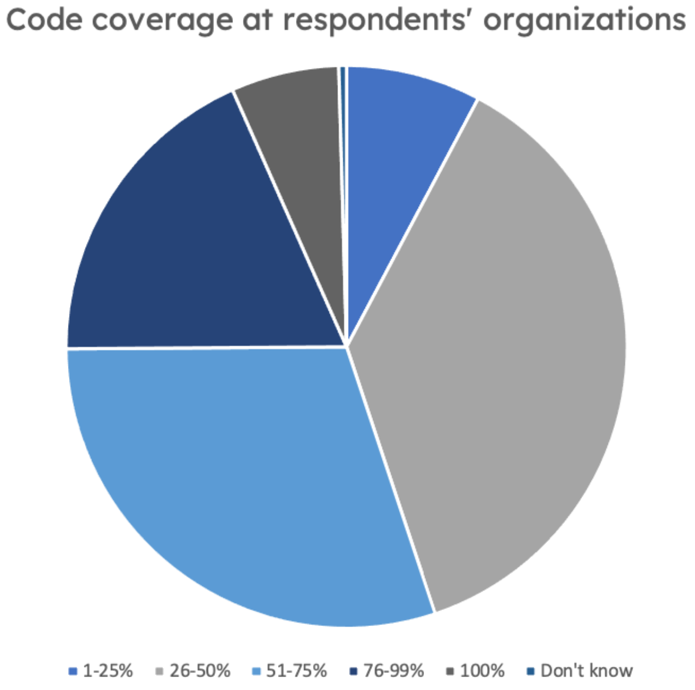

Figure 6: Percentage of organization's Java code currently covered by unit tests

Respondents were also asked how open their organizations are to trying new testing tools, and how open they are to trying new testing tools personally. 94% of respondents overall agreed that their organization is open to trying new testing tools; 95% of SFU said the same, as did 92% of NFU and 84% of NSFU.

When it comes to personal preferences, 96% overall agreed that they are open to trying new testing tools (97% of SFU agreed, as did 96% of NSFU and 87% of NFU).

Code Quality

Because code quality is directly related to testing practices, that link was expected to be reinforced here—and it was. SFU are more likely to say their organization's ability to test code is 'Excellent' (54%, compared to only 44% of NSFU and 39% of NFU). They are also most likely to say their organization's code is of the highest quality (46% of SFU, vs 44% of NSFU and 39% of NFU). In general, however, most respondents in all groups say their organization's code is of high quality (88.9% total).

Respondents had the option to select the top 3 metrics that are most important for deciding how successful an organization's code is, out of: speed, cost, quality, correctness, stability, or 'other'. Among all respondents, quality was selected most often (by 77%), speed was the second most frequently selected (by 64%), and stability was third (selected by 63%), and there were no notable differences between the different groups of framework users.

When asked to describe their organization's code (choose all that apply: reliable, maintainable, testable, portable, automated, and modern), most described it as reliable (selected by 51.1%), modern (50%), and maintainable (40%). Interestingly, despite all of the time developers reportedly spend on testing, only 37% of respondents overall described their organization's code as testable.

Spring Framework users were the most likely to describe their code as maintainable (42%, compared to 31.6% for No-Framework users and 24% for Non-Spring Framework users), and portable (37.5%, compared to 28% for NSFU and 10.5% for NFU). NSFU were the most likely to describe their code as automated (44%, compared to 36.2% of SFU and 26.3% of NFU).

As for the specific benefits of code coverage: 92% of all respondents agreed that unit tests make it easier to modernize legacy code, and agreement was strongest among users of frameworks (93% of SFU agreed, as did 88% of NSFU and 79% of NFU).

Similarly, 92% of all respondents agreed that unit tests make it easier to migrate to the cloud (94% of SFU agreed, 80% of NSFU, and 74% of NFU).

Part Three: Comparative analysis of other organizational priorities between framework users

Respondents were asked to rate the priority of a variety of organizational initiatives and goals: DevOps, Cybersecurity, cloud computing, adopting AI and machine learning, internet of things, and robotic process automation.

DevOps is the highest priority overall among all groups, with 58% agreeing it is a high priority and 35% agreeing it is somewhat a priority. Every Spring Framework and Non-Spring Framework user listed it as a priority (even if a low one); 5% of the No-Framework users said it is not a priority.

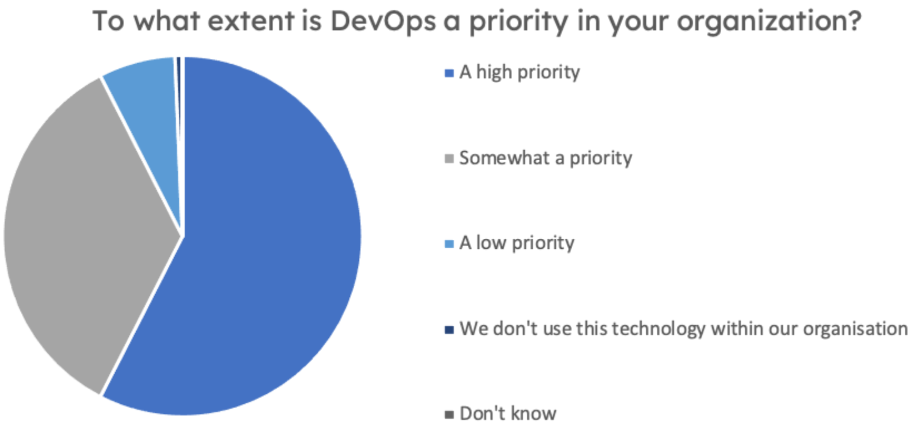

Figure 7: Breakdown of DevOps prioritization

Cybersecurity was the second highest priority overall (57% of all respondents said it is a high priority, and 36% said it is somewhat a priority), and an even higher priority for NSFU (with 68% saying it is a high priority) than the other two groups (56% for SFU and 58% for NFU).

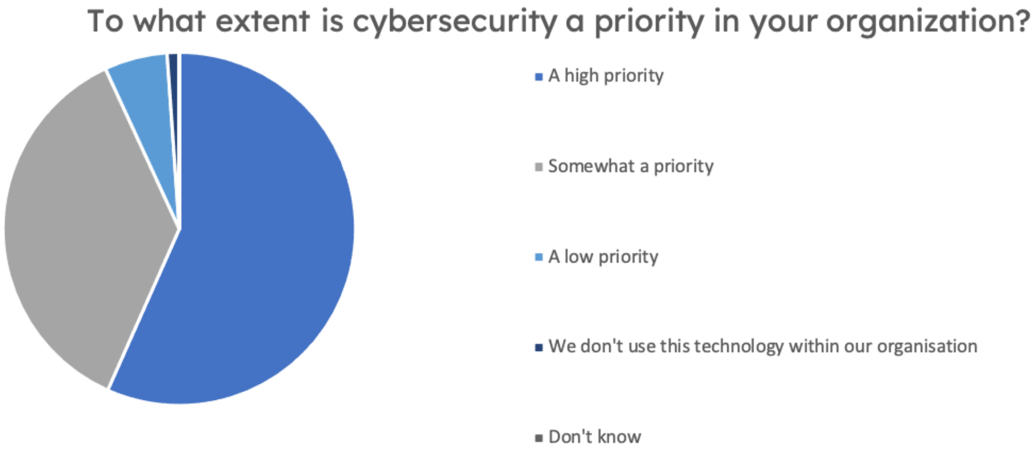

Figure 8: Breakdown of cybersecurity prioritization

Cloud computing came in third place overall (47% of all respondents saying it is a high priority, and 42% agreeing it is somewhat a priority, for a total of

89% of respondents saying it is a priority). It is the highest priority for NSFU (56% saying it is high priority), with SFU in the middle (47%) and NFU at 39%.

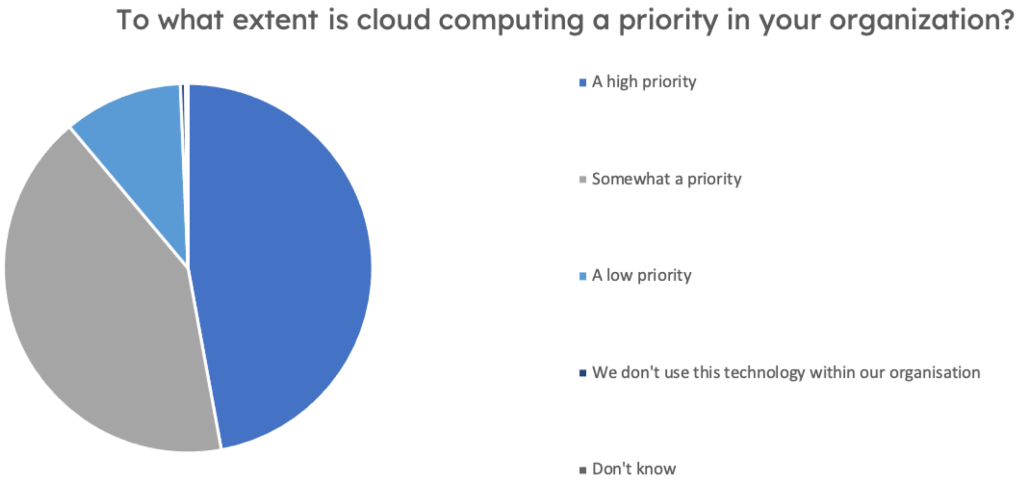

Figure 9: Breakdown of cloud computing prioritization

Adopting AI came in fourth (44% agree it's a high priority, and 43% agreeing it is somewhat a priority); Internet of things was fifth (45% high priority; 45% somewhat a priority); robotic process automation was sixth (40% high priority; 44% somewhat a priority); and adopting machine learning was the lowest priority overall (38% high priority; 48% somewhat a priority).

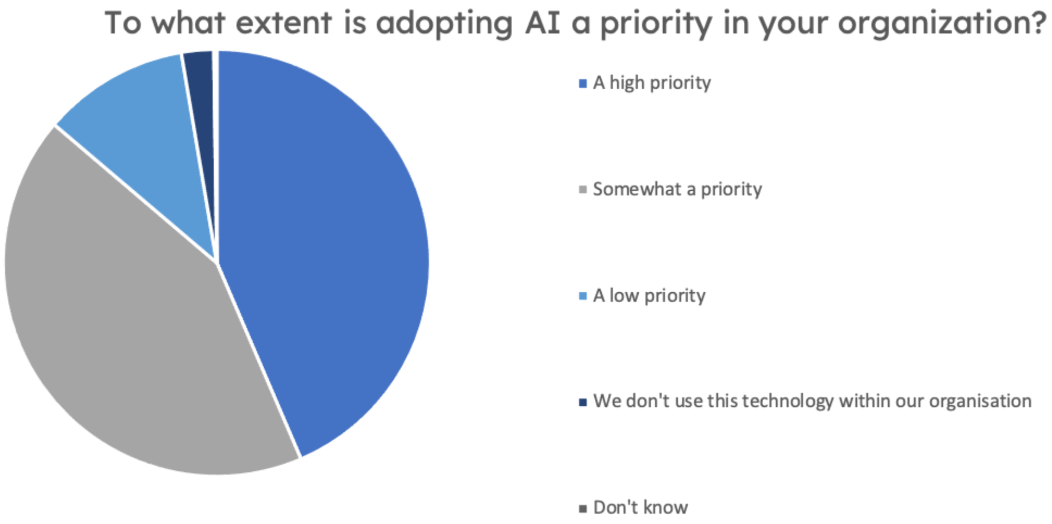

Figure 10: Breakdown of AI adoption prioritization

Discussion and Takeaways

Spring or Spring Boot are used by 86% of the Java developers we surveyed, which is up significantly from the **60% adoption rate** found by a survey collected just last year.

The adoption and use of Spring seem to be led by ease of set up and the time it saves once in use. Spring provides support for testing, which is **a key part of DevOps**—the highest priority initiative among our respondents. The features of Spring that make testing easier also make it better suited for testing tools, which also helps explain why all of the testing tools respondents were asked about were considered to be at least somewhat useful. Our tool **Diffblue Cover**, for example, writes unit tests for Java code automatically. It works especially well for Spring users because of Spring's standardized way of doing unit testing, built-in mocking, and ease of isolating the units under test and database dependencies.

Spring users are also more likely to say that they agree unit tests make it easier to modernize and migrate to the cloud. This could be because features in Spring make unit tests more useful for these activities. In general, however, most respondents agreed that unit testing makes cloud migration and modernizing code easier.

Spring and testing go hand-in-hand

Does Spring help developers be better testers? Or do better testers use Spring? While causation in either direction can't be proven from this survey, there is undoubtedly a correlation between using Spring/Spring Boot and having better tested code. Spring/Boot developers tend to value unit testing more highly, have higher-quality and more maintainable code, and have better code coverage in their organizations, which might be why they gravitated towards Spring in the first place.

But it's also true that having the right tools can make doing the job easier—and Spring makes testing easier. This might be why Spring and Spring Boot users spend 25% more of their time doing unit testing than non-Spring Boot users (25% of their total time, compared to 20%).

While Spring users are the most open to trying new testing tools, most respondents in the other two groups are as well, suggesting that testing is of universal interest, and providing better support for it would be a good idea for all frameworks. Given the amount of time developers spend writing all types of tests (47% of their time, on average, according to our respondents), the popularity of tools that make it easier is not surprising. And as testing becomes of more importance for organizations working towards DevOps goals, maybe it's not so surprising that use of Spring and Spring Boot has continued to rise.