10 BEST PRACTICES FOR A MOBILE APP TESTING PROGRAM

Reaping Optimal Business Outcomes from Mobile Initiatives

INTRODUCTION

Today's mobile Apps are critical elements of customer-facing business strategies. Getting the App right is essential for competitive success, especially with businesses that rely on consumer engagement. As a result, mobile App testing has grown in significance in recent years.

A properly functioning mobile App supports a positive customer experience. Support costs balloon when customers encounter bugs that should have been caught and remediated before release. To help make the most of your mobile App testing process, we've compiled this list of ten best practices. It is based on expert insights from many years of direct experience in this challenging field of testing.

BUSINESS DRIVERS OF MOBILE APP TESTING

Because App functionality is a strategic imperative for business, mobile Apps require rigorous testing. **A bad mobile App experience disrupts the customer's connection with a business.** Any business attempting an omnichannel approach to customer engagement will depend on highly reliable, properly functioning mobile Apps.

Beyond reputational damage, a severe enough App problem can lead to loss of the customer. Customers are costly to acquire and expensive to replace. Excessive customer churn decreases revenue growth. **Effective and thorough App testing helps mitigate this business risk.**

Poorly performing Apps also cause higher support costs. App code defects require rushed patches, notifications to App users and other costly, disruptive practices. Effective testing reduces the likelihood of suffering these financial consequences.

MOBILE APP TESTING OVERVIEW

Mobile App testing is comparable to software testing, but with several important differences. Mobile App testing is far broader in scope. In contrast to general software testing, which might typically test an application on one or two operating systems, mobile testing operates on a much bigger stage. Industry standards dictate that a mobile App must be tested on multiple mobile operating systems and browsers.

Testing a mobile App means evaluating how it performs on different carrier networks. An App can behave quite differently depending on the mobile network. Mobile Apps may also have more connections to third party systems than their PC-based counterparts. Through APIs, a mobile App might need to integrate with data sources and systems in multiple cloud environments as well as on-premises instances.

TEN BEST PRACTICES
FOR A MOBILE APP TESTING PROGRAM

We have performed mobile App testing services for some of the world's largest companies. Our experience spans industries like telecommunications, retail, travel and hi-tech. From this background, we have developed a set of best practices that harmonize to deliver a testing program that exceeds customer expectations.

Mobile testing should be performed on a programmatic basis. Ad hoc mobile testing does not work. There are too many variables to test and too many iterations of code to manage. The practices outlined here work best when they are implemented in a coherent, well-defined testing program. A testing program consists of established testing routines. The program also requires long-term commitment to the testing process.

The goal of these best practices is to help you develop a mobile testing program that meets three critical criteria. **To work, a mobile testing program must be efficient and thorough, but also economical.** Sluggish testing becomes an obstacle to releasing new mobile Apps, leading to a drag on business strategy. Alternatively, frustrated stakeholders may simply circumvent a cumbersome testing process. That being said, the program cannot skip important testing requirements. Missed problems always resurface, repairable at a far higher cost than when dealing with the issues earlier in the development and test cycle.

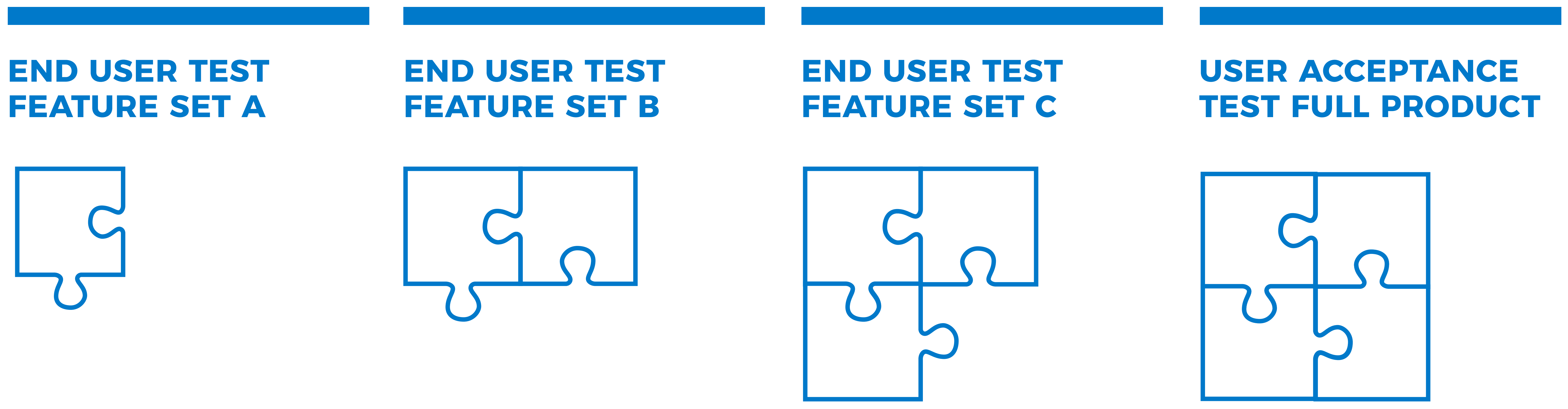
To learn more contact us at
info@mobileintegration-group.com

1 FOCUS ON END USER TESTING FIRST, THEN USER ACCEPTANCE TESTING (UAT)

Best practice conducts end user testing as each feature is developed. Waiting until an App is finished is counterproductive. If there is a problem with a feature, from an end user perspective, it’s better to discover and remediate it immediately.

The key is finding the right end user with whom to do the testing. For instance, if you create a feature for a mass-market consumer mobile commerce App, it needs to be tested by users from that target audience. The end users of the App will likely be people with relatively low technical sophistication.

Conducting end user testing of a mass-market feature with technically advanced fans of mobile gadgets does not result in helpful testing data. In one experience, **we found that end users with low technical sophistication struggled to understand the meaning of icons used in a mobile shopping cart. The experienced mobile commerce users were familiar with the icons and did not notice them in the testing.** It was quite useful to see that for some users the icons were a confusing distraction.


END USER TEST FEATURE SET A
END USER TEST FEATURE SET B
END USER TEST FEATURE SET C
USER ACCEPTANCE TEST FULL PRODUCT

2 | EMPHASIZE SECURITY CRITERIA

STATIC CODE ANALYSIS

A review of the App code that looks for vulnerabilities (i.e. a "Vulnerability Analysis"). This can be done through manual code review, automated review or both.

DYNAMIC ANALYSIS

An assessment of security through the App's real time functionality.

PENETRATION TESTING

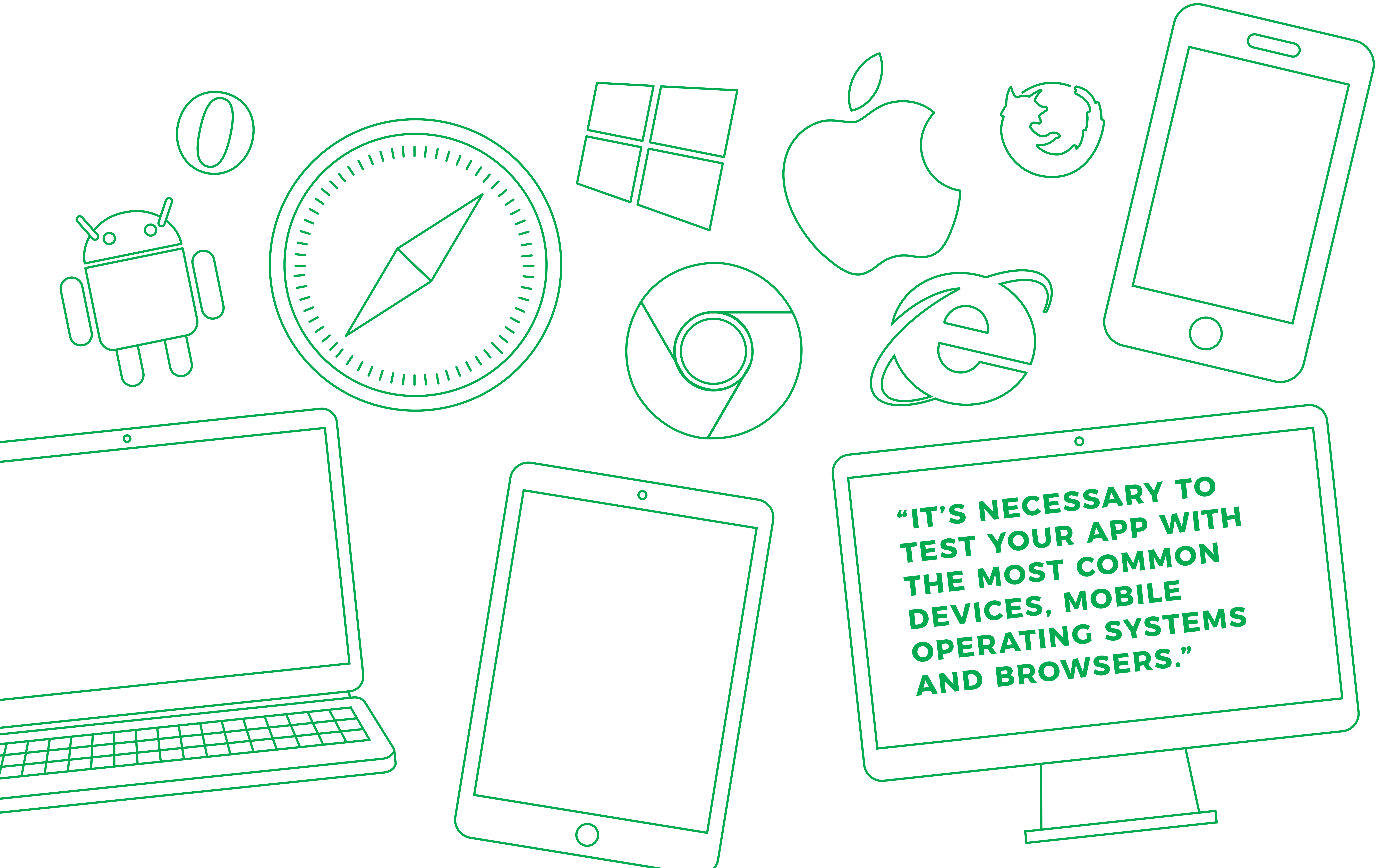
A practical exercise to determine whether it is possible to penetrate the security controls protecting the App.

Mobile Apps expose end users and App publishers to cyber risk. For this reason, **Apps need to contain the security controls and countermeasures required by legal regulations and security policies.** For example, a mobile commerce App that takes credit cards must comply with the PSS-DCI rules that protect sensitive credit card information from unauthorized access. This means encrypting a credit card number used in the App and ensuring that the number has been erased from the phone's memory after it has been used.

Each security measure in an App has to be tested. This may seem daunting, but it is simpler than it sounds. It is not necessary to "Red Team" the App, searching for hidden exploits and performing every security test ever devised. The best practice is identifying the relevant security policies and compliance rules that relate to your App and test for those.

3 | COMPATIBILITY SHOULD COVER DEVICE, BROWSER, LANGUAGE, HARDWARE, OS AND DISTRIBUTED ENVIRONMENTS

Your App should be tested with the most common devices, mobile operating systems and browsers. While it is nearly impossible to test for the tens of thousands of device/OS/browser combinations in use, it is reasonable to test for the industry standard sets. **The standard practice in the telecommunications field is to test an App on five major operating systems and browsers.**

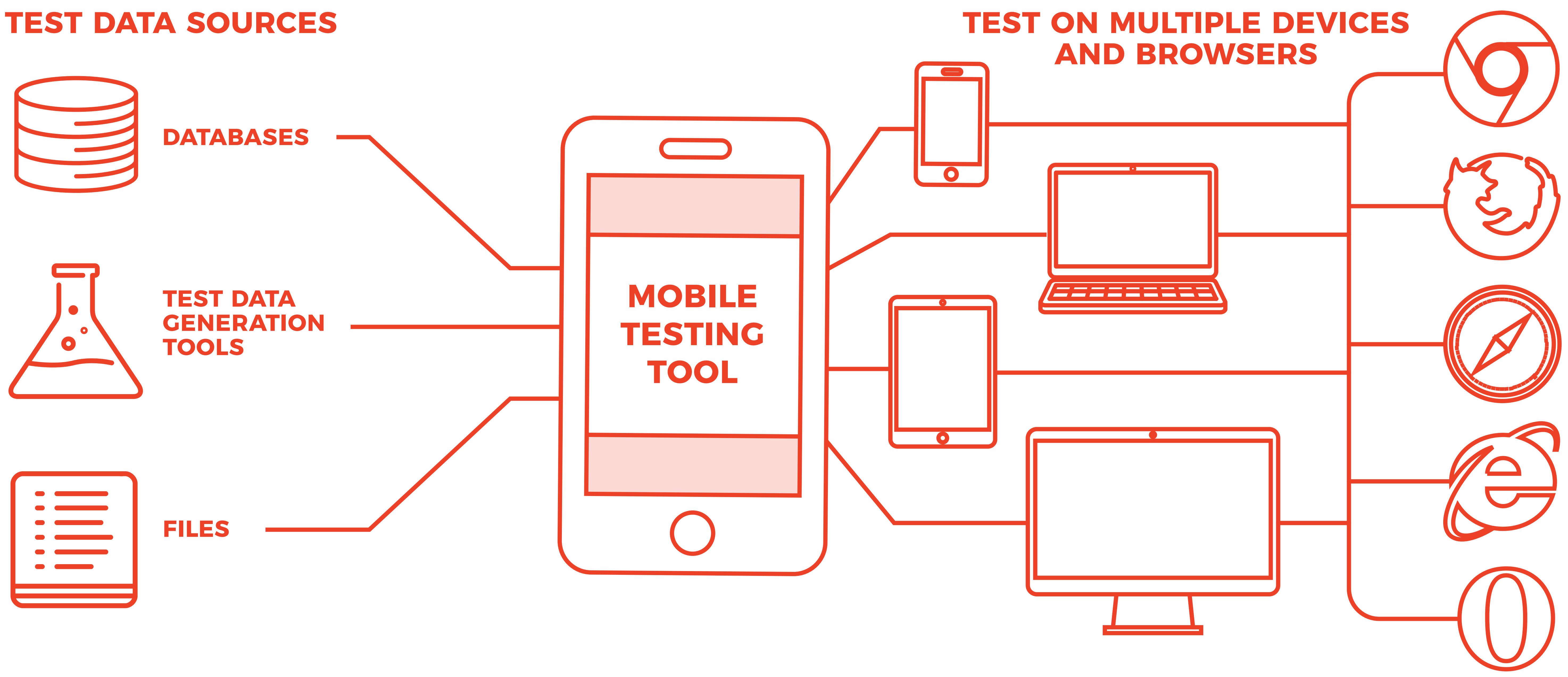

"IT'S NECESSARY TO TEST YOUR APP WITH THE MOST COMMON DEVICES, MOBILE OPERATING SYSTEMS AND BROWSERS."

4 | TAKE A DATA-DRIVEN APPROACH TO TESTING

We recommend a data-driven approach to testing. **The best practice is to develop a representative test data set that demonstrates performance with realistic inputs.** It should be preset, not manually created on the fly as testing progresses.

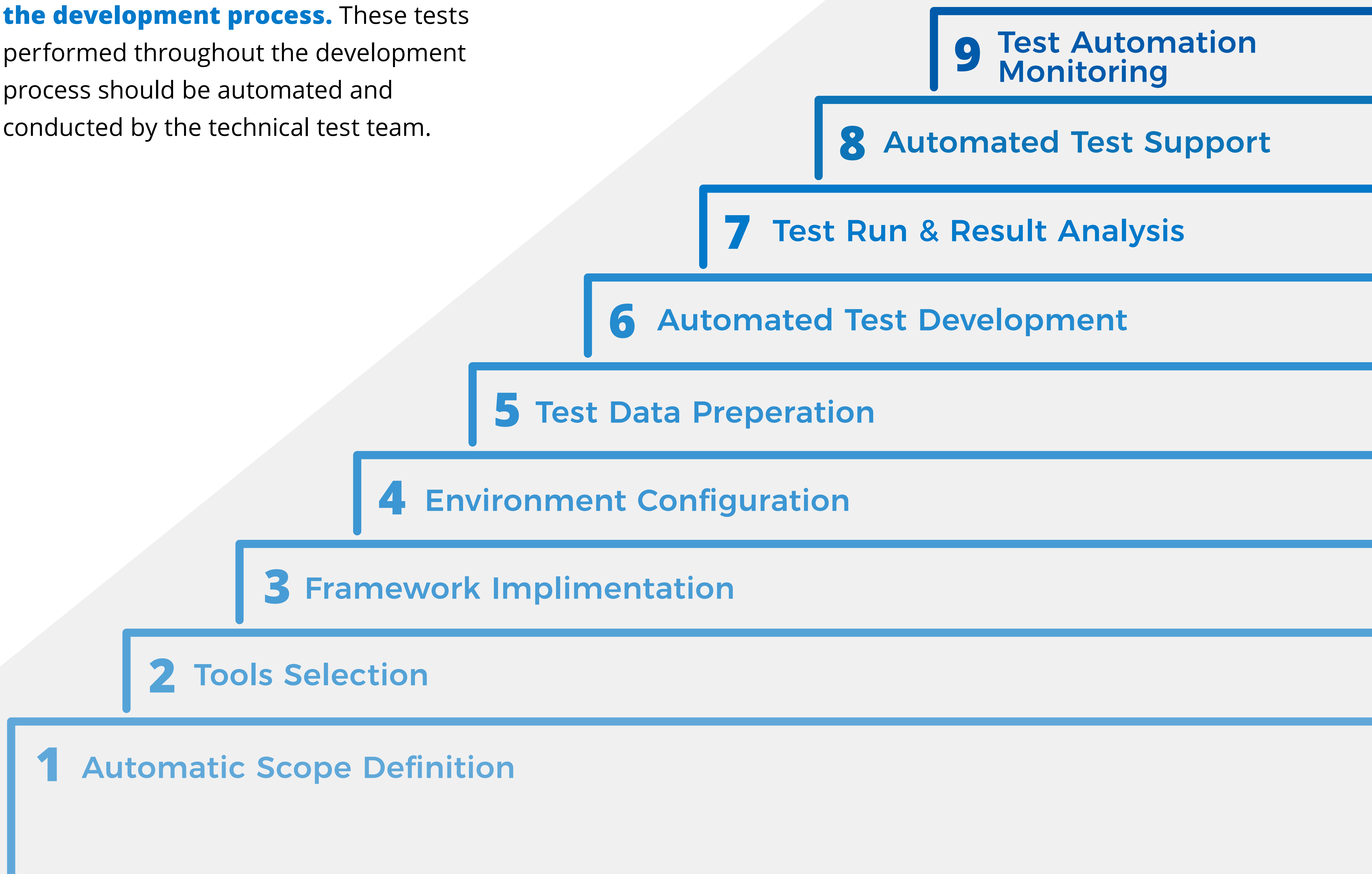
The test data set has to be heterogeneous enough to mimic real life. For example, a mobile commerce App shopping cart should be tested with the kind of variety and quantities of products that actual users will order. Otherwise, the App may "pass" the test but still be deficient.

We recommend using a test data generation tool. You can configure these tools to produce large quantities of test data that align with your case. Some of these tools are able to take production data and mask sensitive, e.g. personally identifiable data from the test set. The advantage in this approach is that you can use the most realistic data set—the data you're already using in production.


TEST DATA SOURCES
DATABASES
TEST DATA GENERATION TOOLS
FILES
MOBILE TESTING TOOL
TEST ON MULTIPLE DEVICES AND BROWSERS

5 TEST AS YOU DEVELOP

App owners don’t like surprises. You can avoid surprises by testing your mobile App code as you develop it. This is a two-step process. **First, you have to understand what functionality you will test at each stage of development.** Testers should coordinate with developers to map out the functions that will be developed at each stage of coding. **Then, you must develop automated test scripts and get your test data sets in line with the development process.** These tests performed throughout the development process should be automated and conducted by the technical test team.


9 Test Automation Monitoring
8 Automated Test Support
7 Test Run & Result Analysis
6 Automated Test Development
5 Test Data Preperation
4 Environment Configuration
3 Framework Implimentation
2 Tools Selection
1 Automatic Scope Definition

6 | TEST FOR OBJECT-ORIENTED CODE SCENARIOS

Object-oriented code for mobile App development may result in hidden defects in the final product. We advise testers to implement the following object-oriented test scenarios:

Inheritance – Objects may be based on another object or class resulting in the inheritance of conflicting functionality. Inheritance is problematic when the functionality does not match the App's requirements.

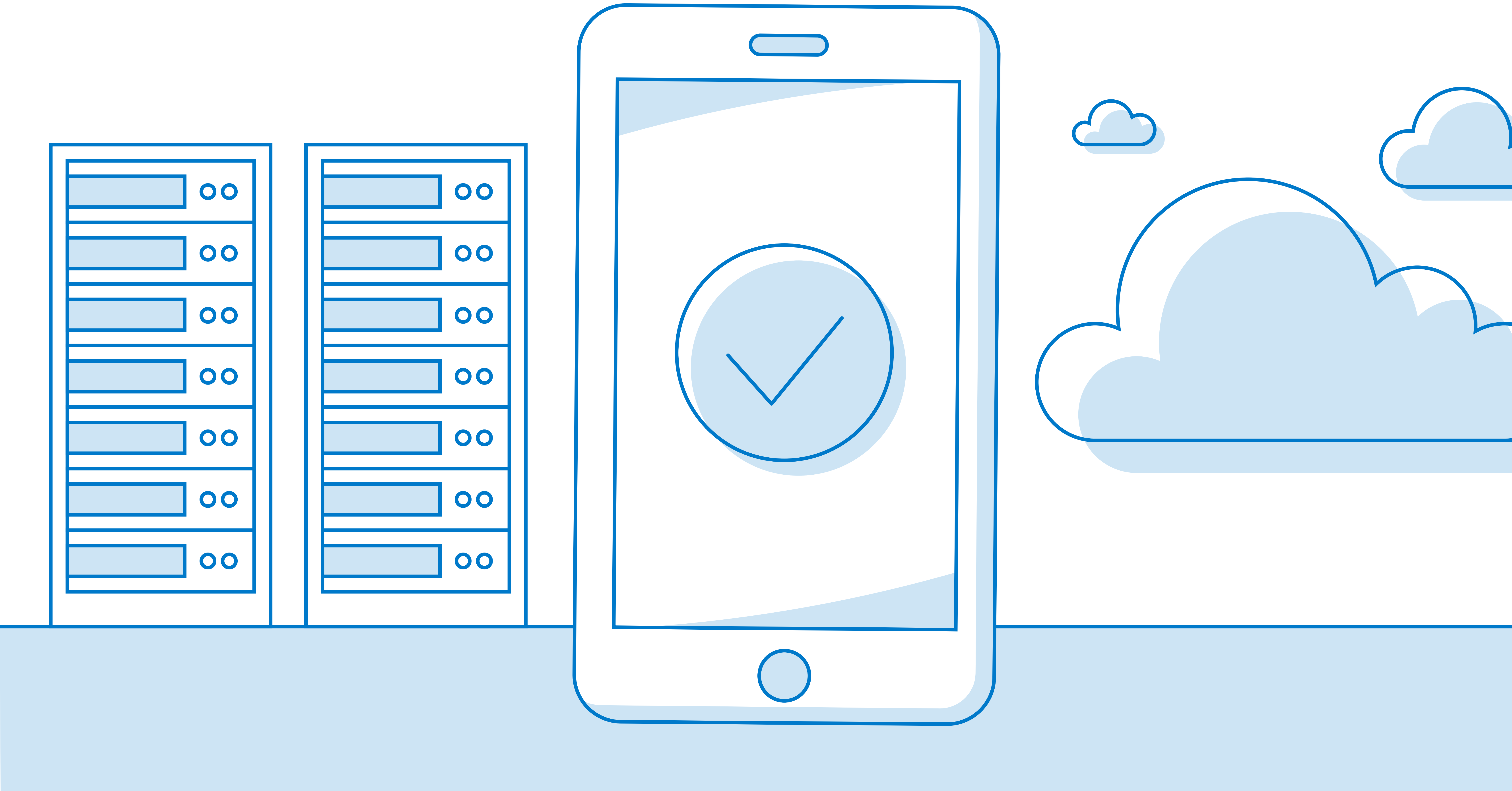
Dependencies – Objects often work in tandem with other objects, creating dependencies in the code. Testing helps you avoid situations where an object seeks the functionality of a dependent object but does not find it.

Abstraction – To reduce duplication of code, object-oriented developers may abstract multiple functions behind a single object. If not thoroughly tested, abstraction may result in unpredictable app behavior or failures.

Encapsulation – An object may hide its internal state from other interacting objects, a concept known as encapsulation. Encapsulation masks the true functionality of the object. Only testing verifies whether the object is performing as expected.

7 | INCLUDE ON-PREMISES, HYBRID AND CLOUD TESTING SCENARIOS

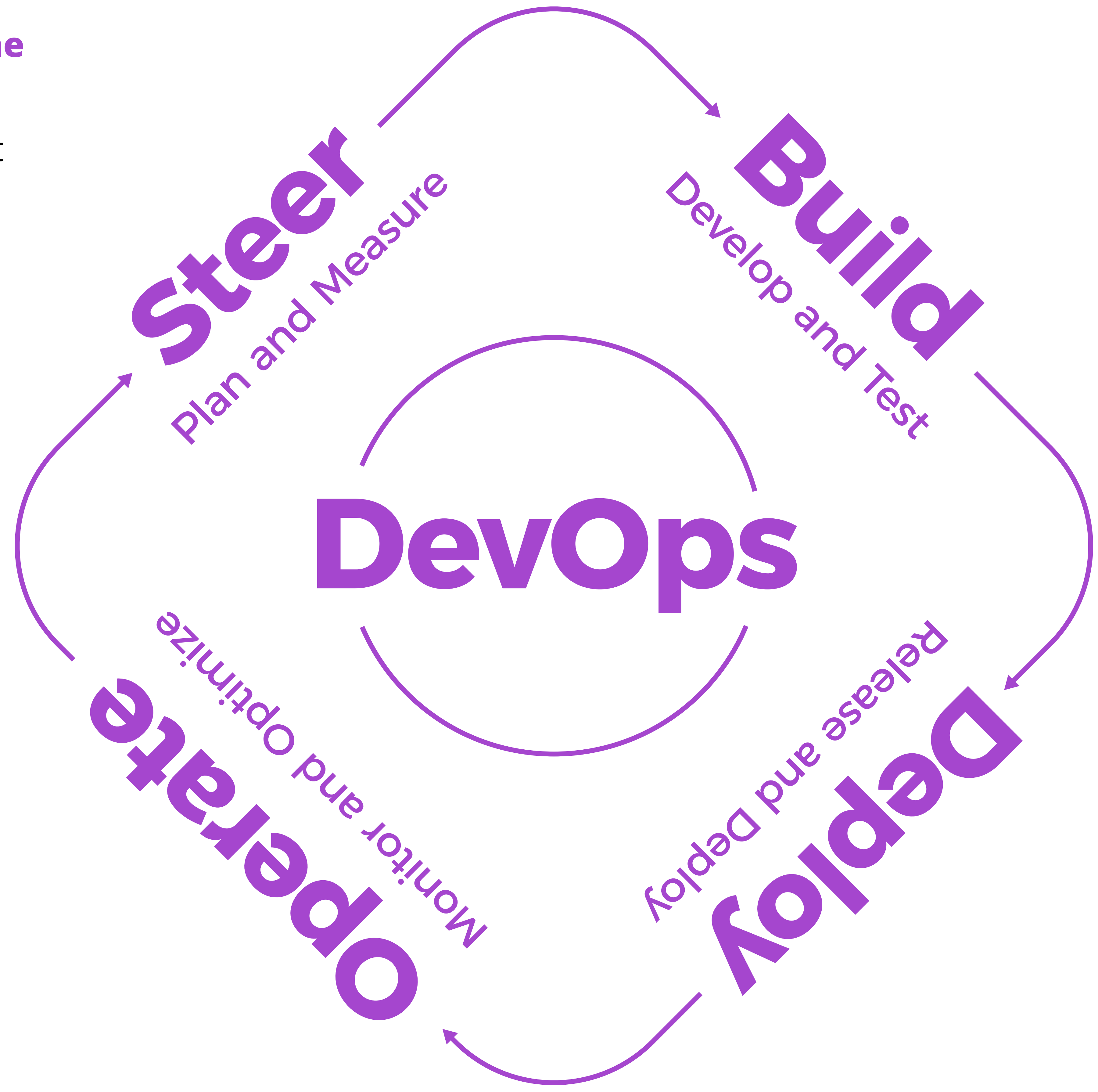
Mobile Apps routinely interact with on-premises, hybrid and cloud-based servers. **An effective testing regimen should gauge how well the App functions in all three server-hosting scenarios.** We recommend starting with emulators and mobile simulators. Then, after testing App functionality, performance and security with on-premise, hybrid and cloud servers, test again using physical devices.



8 | EMPLOY THE SPRINT MODE OF TESTING

Testing should align with the App's development methodology. Most developers of mobile Apps today use an agile/scrum methodology of some sort. This approach consists of "sprints," where the coding team develops a set of features before moving on to the next sprint. **The goal is to break the testing process down into pieces. Waiting until all the development is finished before testing will needlessly elongate the entire development and test cycle.** This delays time to market and typically requires extra development and test cycles to repair problems that could have been identified much earlier.

Best practice in this case is testing the features developed in a sprint right after the sprint concludes. That said, it is important to finish testing before the next sprint is initiated. One possible enhancement of this approach is to adopt a phased sprint, with options to pivot at any point on what is tested versus waiting.



9 | USE AUTOMATION TOOLS SPECIFIC FOR EACH MOBILE PLATFORM

The testing tool market offers many options. It pays to match testing tools with the platforms being tested. Some work better with iOS, others with Android and so forth. For instance, one tool might pull data from the App differently from another. This could have an impact on testing results. **Additionally, if an App is tested on behalf of a client, it is wise to ask them if they prefer one testing tool over another.** Getting clarity in advance will save time and aggravation later.

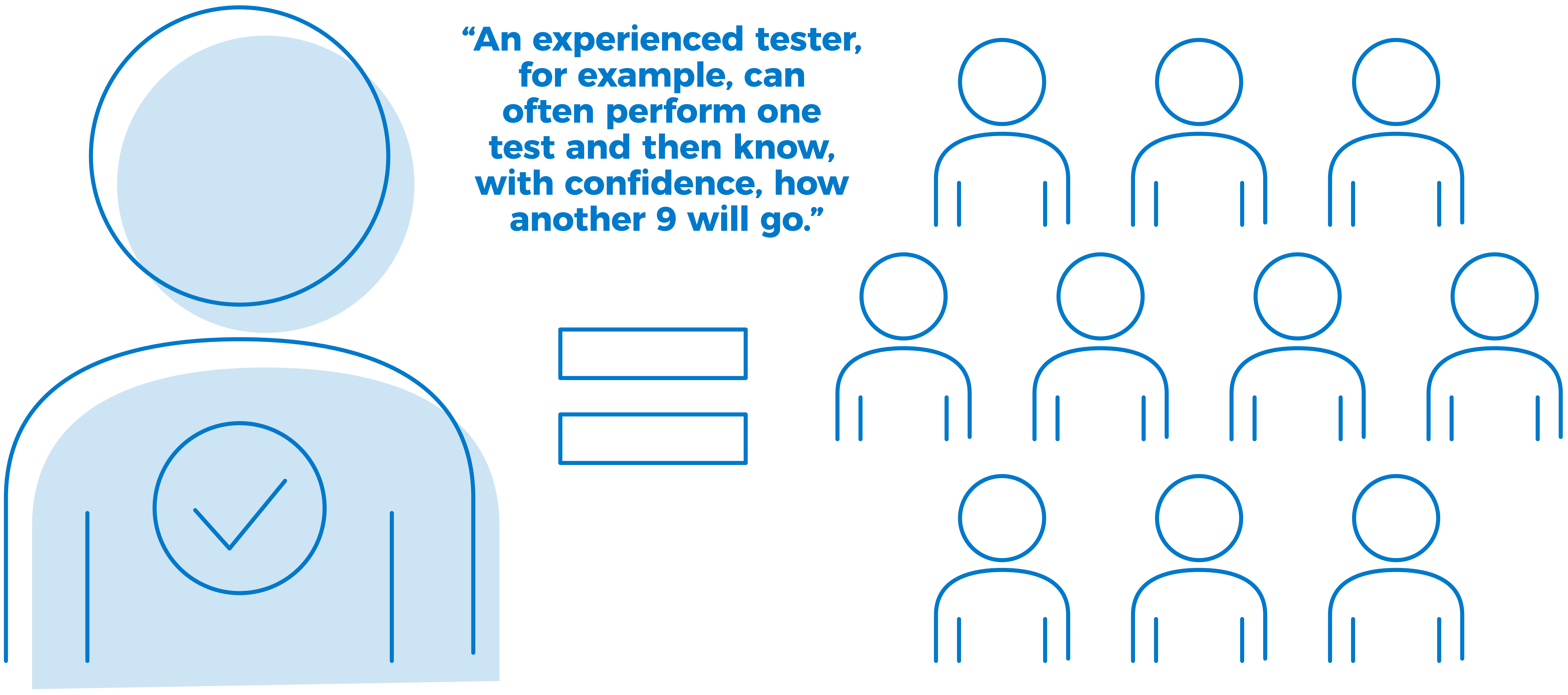
"It pays to match testing tools with the platforms being tested."

10 | DON’T COMPROMISE ON EXPERTS AND AVOID THE FALSE ECONOMY OF LOW COST LABOR

Testing is potentially rife with wasted effort. Certain tests are obligatory, but testing should be focused on quality, not quantity. Some testing organizations deploy lower-skilled testers to perform tests that are either unnecessary or improperly carried out. **Being offered 300 tests is not automatically better than getting 100 performed by a tester with deep experience.**

An experienced tester, for example, can often perform one test and then confidently predict how another nine will go. The less experienced tester might perform all ten just to be sure. That is a waste of time. It is also a false economy. **Higher skilled testers usually cost more, but an investment in quality is worthwhile in the vast majority of testing scenarios.**

“An experienced tester, for example, can often perform one test and then know, with confidence, how another 9 will go.”



CONCLUSION

Mobile Apps are critical to the success of your business. To ensure proper functioning, you need a **thorough, efficient and cost-effective** approach to App testing. Based on our experience, we have devised a set of best practices to help you develop a mobile testing program.

We recommend focusing on **end user testing** first, then **User Acceptance Testing** (UAT). Security criteria are essential, especially in today's ominous environment. **We advise testing for compatibility across device, browser, language, hardware, OS and distributed environments.**

A **data-driven approach** keeps the testing process aligned with your business requirements by evaluating the App using an appropriate test data set.

Other best practices involve **matching testing to the App development methodology, testing as you develop and conducting testing sprints**. It pays to test for a full range of object-oriented coding scenarios like aggregation and encapsulation. **Testing should address differences between on-premises and cloud hosting environments.** Use the automation tools best suited to each mobile platform. Finally, work with experts. You'll avoid wasting time and money on unnecessary and often inconclusive testing. By adopting these practices, you'll be in a position to implement an effective mobile app testing program.

To learn more contact us at
info@mobileintegration-group.com