SHIFT LEFT TESTING

Shifting Left to Keep Pace in Mobile and IoT Testing

WHITE PAPER
SHIFT LEFT TO KEEP PACE IN MOBILE AND IOT

Speed and Quality Required

In the world of mobile and IoT, enterprises face immense pressure to move faster and more flexibly in application development and testing – without sacrificing anything in innovation, product quality or service levels. Any compromise risks dissatisfied customers, reduced revenues, and competitive disadvantage.

To this end, many organizations are moving aggressively to automate their testing and QA processes to enhance efficiency and expand the scope of their quality efforts. However, without the right resources, tools and expertise, this can be extremely difficult. These challenges are only exasperated by the plethora of different platforms, devices, sensors and operating systems to test against.

"Shift Left" is a recent movement that is helping enterprises address both the speed and quality requirements created by the new world of mobile and IoT. The basic principle driving Shift Left is applying more QA efforts earlier in the software development lifecycle (SDLC) to identify potential problems when they are easier and less costly to fix. This is achieved by enabling QA to work in parallel with development during the requirement and design stages of the development lifecycle.

This paper discusses some of the key considerations for bringing effective Shift Left approaches to your testing and QA efforts and the potential business impact for your organization.

Shift Left – Moving Quality Forward

By moving quality considerations to the start of a development project, Shift Left enables QA teams to find bugs and defects sooner and development teams to fix them more quickly and less expensively – before they impact a customer. This model ensures quality throughout the entire software development lifecycle, reduces costs, accelerates release cycles and increases customer satisfaction.

Common Quality Compromises

QA and test management have become very complex in the digital world with the numerous platforms, OSs, devices, use cases and variety of network connections and carriers involved. Mobile, web and hybrid apps have to be tested across multiple permutations and combinations. In our experience with both market leaders and emerging innovators, we see organizations come up against several common challenges as they try to ensure quality products and services in mobile, web and IoT.

DYNAMIC PRODUCT REQUIREMENTS

- Fast changing requirements demand additional features in existing applications
- Interdependencies between features introduce unforseen application errors

COMMUNICATION GAPS

- Implicit requirements leading to incorrect assumptions and misinterpretation
- Miscommunication about requirements
- Complex business requirements that are not fully understood

ERRORS DURING DEVELOPMENT

- Developers not understanding requirements fully
- Assuming too much on their own
- Shortcut methods to deliver application in short amount of time

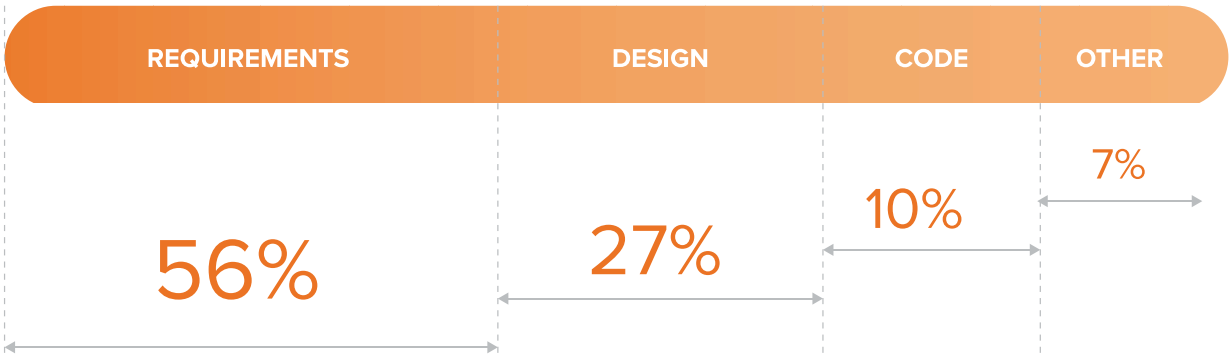
INSUFFICIENT TESTING & QA

- Incomplete testing against client requirements
- Compressed time windows require tradeoffs
- Quality not meeting the standards

Impact of Defects

According to a recent study by quality consultant SQS AG, 56% of software defects originate during the requirements phase of a project, and 27% in the design stage. This compares with only 10% that originate during the development (coding) phase.

% Of Bugs Originating During These SDLC Stages

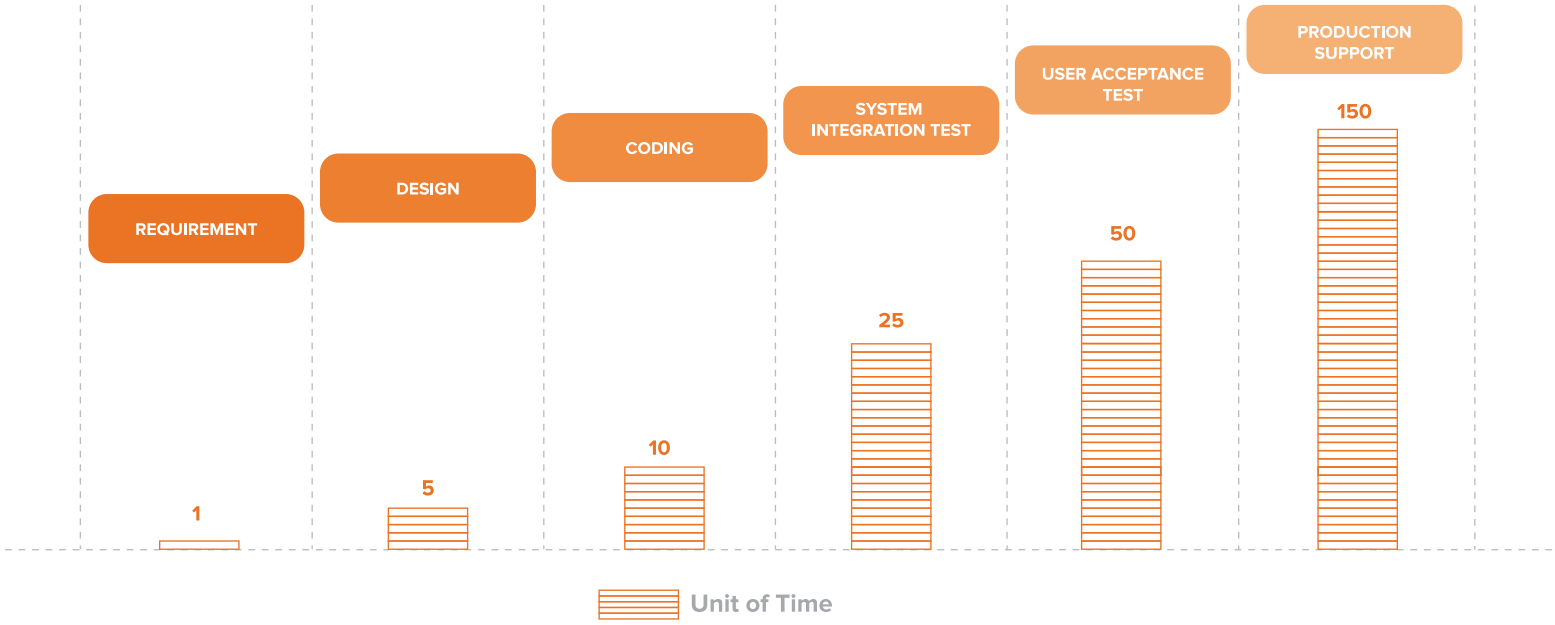


Impact of Bugs

In traditional software development and test processes, testing is initiated at the coding stage – when 83% (56% + 27%) of the bugs have already been introduced. The impact of this is significant – both in terms of time to resolution and cost.

Time to Resolve Bugs/Defects

A recent study titled "The Journal of Defense Software Engineering" by CrossTalk, measured the difference in the time it takes to resolve an issue/defect found at the early and late stages of the software development lifecycle. The study determined that it can take as much as 150x longer to resolve a defect found in production vs. one found at the requirements stage; and 30X longer than one found in design.

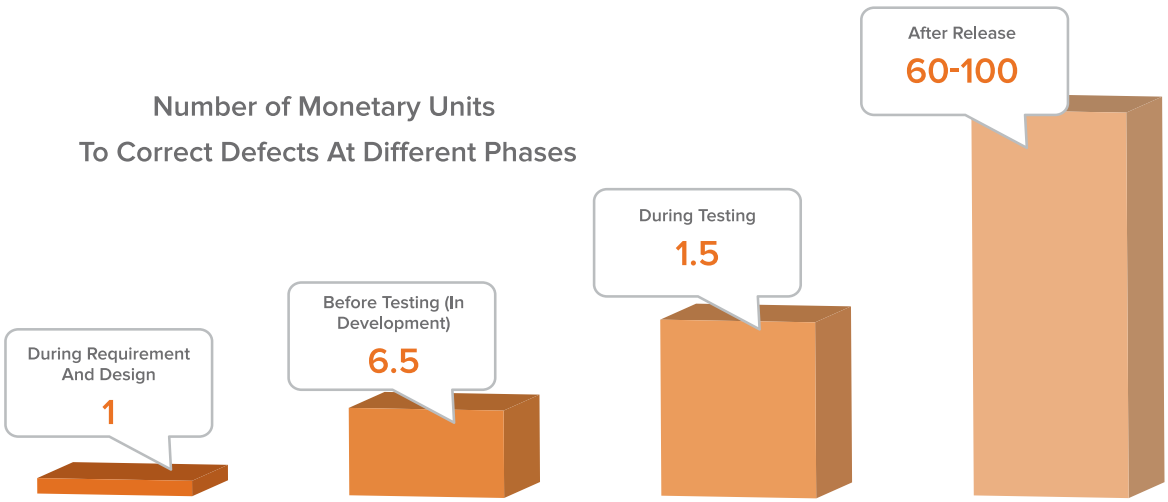


With most testing and QA processes not beginning until the coding stage, companies find it much harder to respond, resulting in:

- Extended testing cycles that slow development and delay time to market
- Increasing testing and QA costs
- Poor customer experience and increased complaints as a result of defects
- Higher customer service costs

Cost to Resolves Bugs/Defects

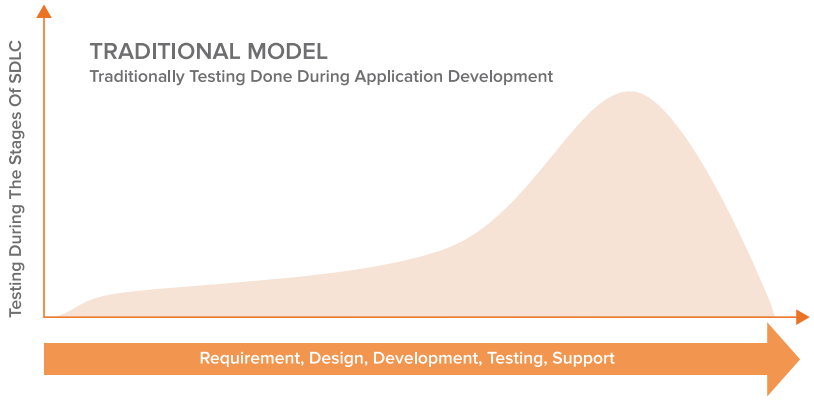
IBM Research also looked at the relative cost to correct bugs and defects based on when in the software development lifecycle they were identified and resolved. The study showed that defects identified and resolved during the Requirements and Design phase of development are 60-100X less expensive to fix than those discovered and fixed after product release.



(Defects introduced during the requirements and design phase are typically more severe and therefore more difficult to remove via testing.)

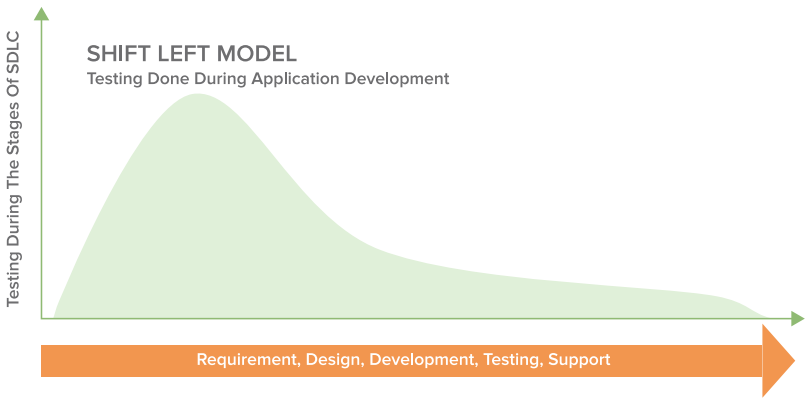
Shifting Left

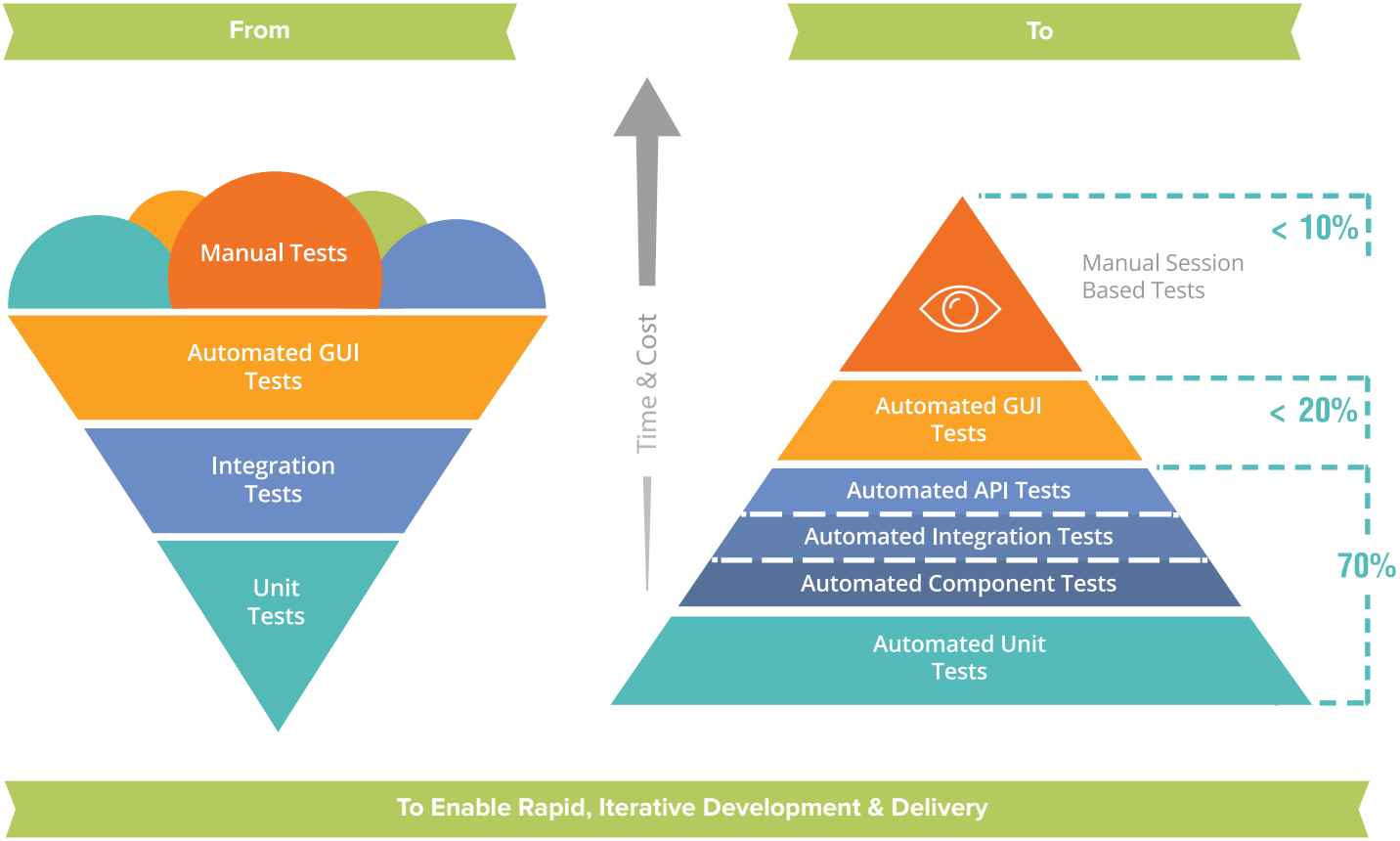
In software development, prevention is always better than a cure. To develop high quality applications at a competitive pace, Test/QA teams must find and resolve defects earlier in the development process; i.e., they must "Shift Left."



Traditional testing models rely on a sequential software development process model that follows pre-set defined phases; e.g., analysis, design, coding, and testing. In this kind of traditional testing procedure, the main focus is on the back end of the process vs. the Unit testing performed earlier by developers using automation. As a result, most defects are not identified until late in the development lifecycle when they are more difficult and expensive to fix. The testing is very manual and inefficient and also very limited from a coverage point of view.

By contrast, Shift Left focuses on testing earlier in the development lifecycle, allowing developers to find errors when they are much easier to address. This reduces the time between releases, improves software quality and delivers new features that increase customer satisfaction, faster. Most organization have automated only a small percentage of their test footprint. Shift Left can help flip that paradigm such that manual testing is the exception in your organization.





Source: watirmelon.com

Infostretch and "Shift Left"

At Infostretch, we have extensive experience in helping organizations "Shift Left." Leveraging proven best practices, tools and frameworks, we optimize testing and QA and embed Shift Left concepts and procedures into the application development lifecycle with the goals of "Prevention" and "Prediction."

This includes a well-defined and structured defect prevention methodology shared by project implementation teams to identify, analyze and prevent bugs and defects throughout the development lifecycle. Standardizing and automating these processes across the team creates greater predictability in identifying defects and our extensive experience in the field of testing gives us access to a large base of knowledge to prevent potential defects before they happen.

Our approach to Shift Left encompasses the end-to-end software development lifecycle.

Testing

Shift Left requires teams to start test execution early in the project development lifecycle. The testing team should have a mix of two types of resources:

- **Test engineers** – team members who are adaptable to technology as well as possess business domain knowledge to facilitate the testing.
- **Test developers** – team members who are technically strong and can develop automated test scripts. This team will develop scripts, perform code reviews and have automated test cases ready to be used.

Development

In Shift Left environments, development team needs to follow a three-step process:

- **Prioritize feedback** – As the testing team conducts test and provide feedback in the early stages of the SDLC, the development/coding team needs to prioritize the feedback based on the product requirements as well as issue severity.
- **Process and acceptance** – based on above
- **Resolution** – Make team resources available to act on priorities and resolve them.

Allocation

Allocate or free enough resources to provide the feedback as well as to act on the feedback provided within the stipulated amount of resolution period

Deployment/Automation

Automation is critical to the Shift Left approach – from deployment of virtual services, to installation of the latest build, the whole process should be automated. For example – Develop a task that will trigger performance automation. This automation will gather performance related data, match this data against the benchmarked results and flag any of the highlighted or risky areas in the reports.

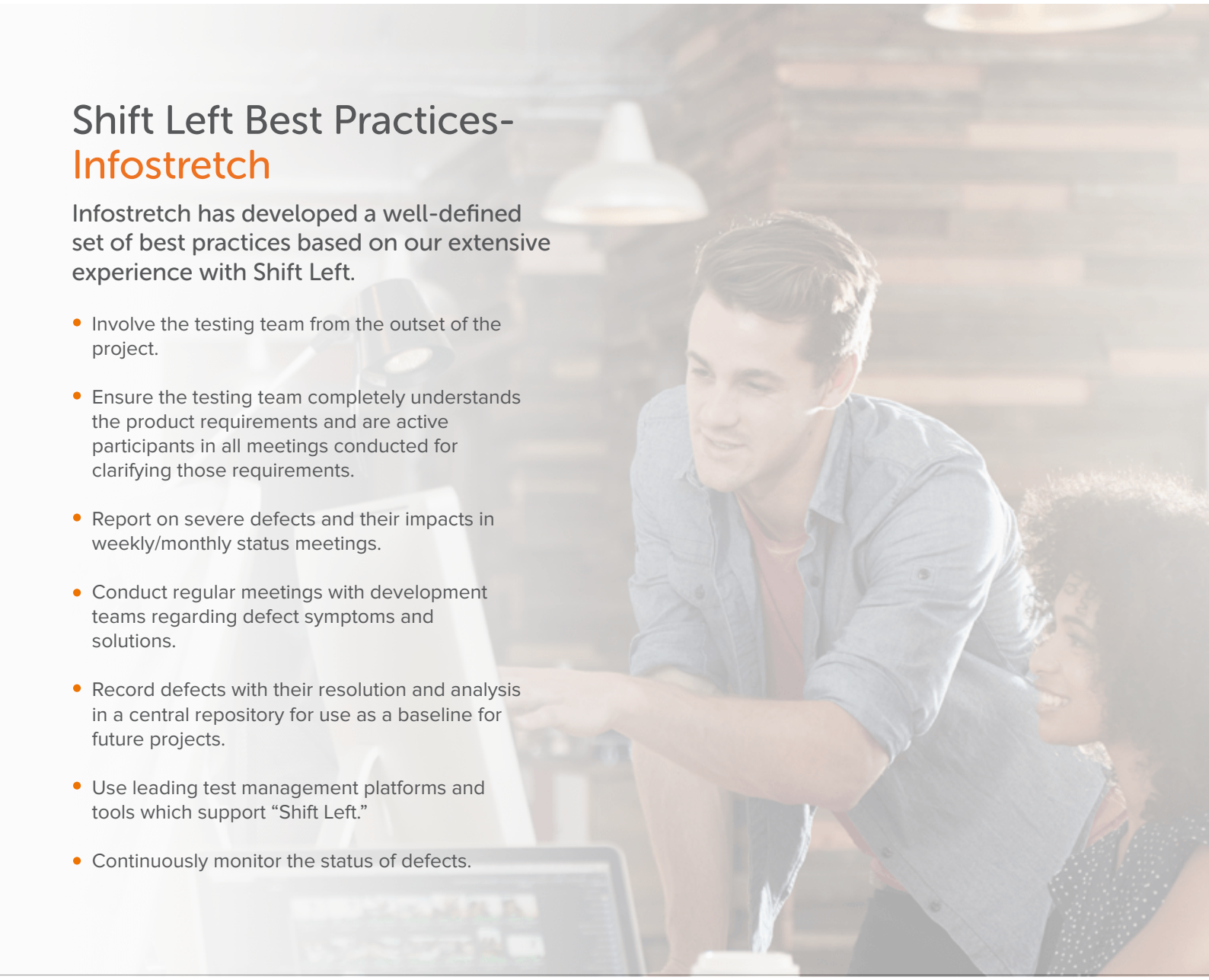
Infrastructure

Infrastructure should not be overlooked. In many cases, your team will be prepared and available to conduct early testing, but the test environment will not for any number of reasons. This can create big inefficiencies. To resolve this, plan, create and virtualize the services wherever needed.

Consider the following:

Agile projects typically involve multiple development teams working on different parts of the same project. These teams may not all deliver their piece at the same time. For example, let's say there is a mobile application being developed along with the backend system that supports it. In this scenario, there will be two teams – one responsible for developing the mobile application front end, another responsible for developing the backend; i.e. database, server configuration, etc. In this case, the front end team may complete their development very early in the process; well before the backend team finishes their piece.

In this situation, the front-end team may have to wait for the other team to complete the development before they can start with the testing. To cope up with this kind of situation, test virtualization can be done. This allows the front-end team to simulate the back-end system. At the same time test virtualization will keep the code changes from the back-end team in synchronization. This keeps the project moving without having to wait for the backend environment to be available.



Shift Left Best Practices- Infostretch

Infostretch has developed a well-defined set of best practices based on our extensive experience with Shift Left.

- Involve the testing team from the outset of the project.
- Ensure the testing team completely understands the product requirements and are active participants in all meetings conducted for clarifying those requirements.
- Report on severe defects and their impacts in weekly/monthly status meetings.
- Conduct regular meetings with development teams regarding defect symptoms and solutions.
- Record defects with their resolution and analysis in a central repository for use as a baseline for future projects.
- Use leading test management platforms and tools which support "Shift Left."
- Continuously monitor the status of defects.

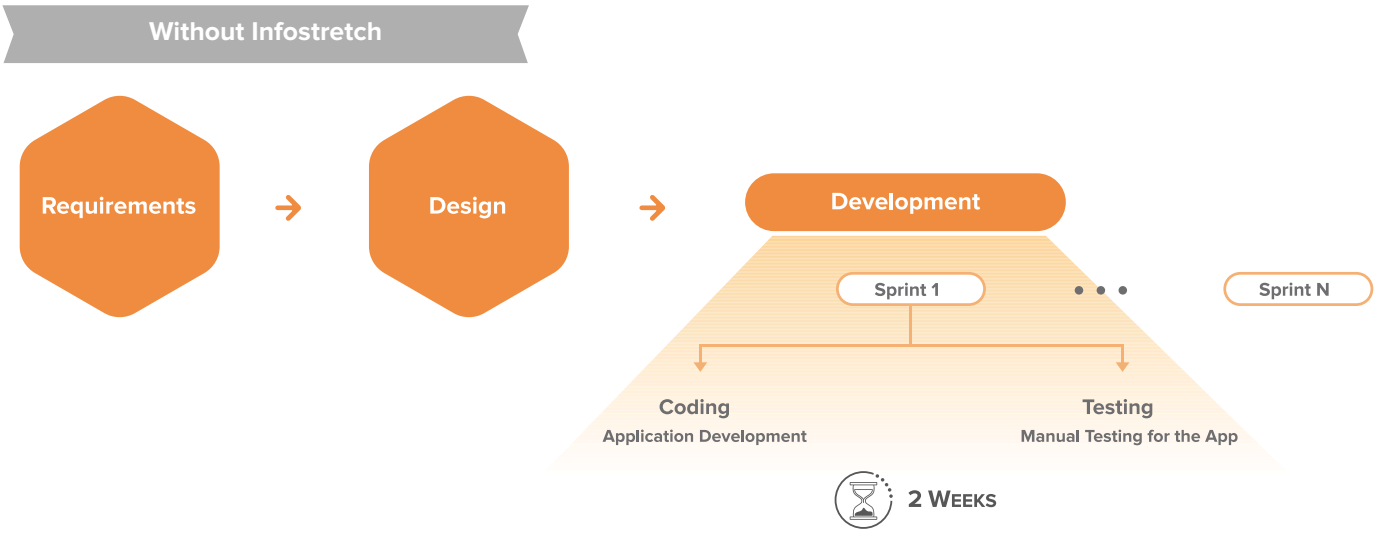
Automation is also key

In addition to the best practices outlined above, Shift Left also depends on automating manual test/QA processes wherever possible. This requires identifying:

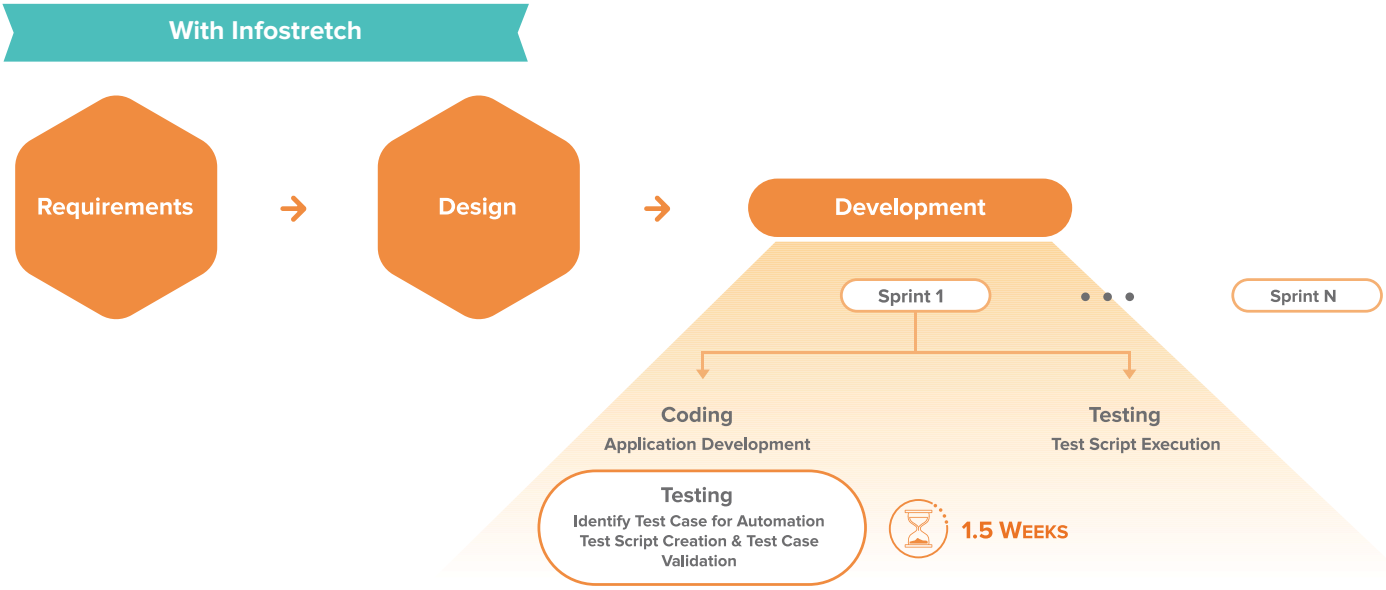
What to automate?	How to automate it?	When to automate?
Which tests, devices, geographies and use cases?	Which tools, managed services, integrations?	At what point in the QA cycle, are processes repeatable; are use cases stable?

The importance is illustrated in the two diagrams below

In the traditional Agile approach, "Without Infostretch" below, there is a linear process starting with requirements and continuing through design and then development. User stories are defined, sprint planning takes place, sprint execution of a periodic time – let's say for example a sprint of two weeks is started. This two-week sprint contains two weeks of application development and two weeks of testing in parallel.



In the Shift Left Agile example below labeled "With Infostretch," the testing and automation teams are involved from the start of the project. So, they fully understand the requirements. Now when development starts, the test and automation can identify the test cases for automation, create the test scripts and also validate these test scripts for automation – all in parallel. Testing is accelerated because only execution of testing scripts needs to take place. As a result, testing cycles can be completed with 1.5 weeks, a 25% reduction.



Shift Left accelerates business

By embracing Shift Left, companies can identify and correct defects as they occur, speeding cycle time, reducing costs and increasing customer satisfaction and service levels. Shift Left does not happen on its own though. It demands accountability and increased communication with all stakeholders across the software development lifecycle. But the results are worth it. At Infostretch, we have seen organizations reduce overall testing cycles from 1-2 weeks to 2-3 days by implementing the Shift Left methodology with Agile testing and increased automation.

Accelerate your digital initiatives with Infostretch.

Contact our team today for a free consultation.

Infostretch helps enterprises rapidly launch new mobility and connected product initiatives, with greater success and less risk. Infostretch is an expert in implementing Continuous Integration (CI) and Continuous Delivery (CD) methodologies and tools and transforming businesses from Quality Assurance (QA) to Quality Engineering (QE).

infostretch

408.727.1100
info@Infostretch.com
www.Infostretch.com

Infostretch Corporation. 3200 Patrick Henry Drive.
Suite 250. Santa Clara. CA 95054