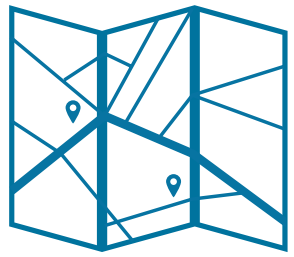
QUICK AND EFFICIENT

MOBILE TESTING STRATEGY

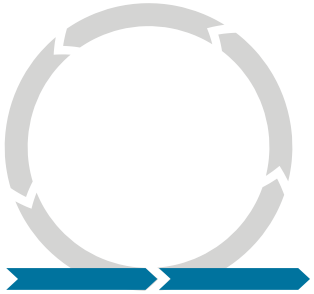
ABOUT CHAIONE

ChaiONE is an enterprise mobility agency focused on creating innovative, beautiful mobile solutions that solve complex business problems of large enterprises.



MOBILE STRATEGY

USER EXPERIENCE DESIGN



AGILE DEVELOPMENT

QUALITY & ASSURANCE

WRITTEN BY

LAVANYA SUKUMAR

Lavanya Sukumar is the Quality Manager at Chaione. She is responsible for creating Chaione's test framework and strategy document, testing both mobile and web applications across various platforms, conducting UI/UX reviews and defect analysis for all engagements.

@lavanyasaikumar

DESIGNED BY

KAITLEN PERKINS

Kaitlen Perkins is an Interaction & Visual Designer at Chaione. She is responsible for crafting interactive user experiences by utilizing research, prototyping, and reinforcing visual aesthetics.

@kaitlenmax

DESIGNED BY

OSAMA ASHAWA

Osama Ashawa is an Interaction & Visual Designer at Chaione. He is responsible for crafting interactive user experiences by utilizing research, prototyping, and reinforcing visual aesthetics.

@lmaosama

QUICK AND EFFICIENT MOBILE TESTING STRATEGY

Without Compromising on Quality and Overcoming Fragmentation Issues

01
Introduction

INTRODUCTION

The need for a quick and efficient test strategy has become an integral part of the Agile framework. Quality analysts (QA)/ testers, or "internal clients" make and break a product's reputation. This important role emphasizes the need to spend enough time and test every aspect of the product in detail.

The question here is,"Do we get that sufficient time to test?" Every quality analyst would say"NO." We can definitely try to estimate test efforts, plan well in advance, allocate sufficient time for regression testing and freeze the code before regression.

But can we implement all of these within the given deadline? In such situations, it is the responsibility of the quality analyst to test the applications quickly, efficiently and without compromising quality. He should also try to use as many devices as possible and aim for ontime delivery.

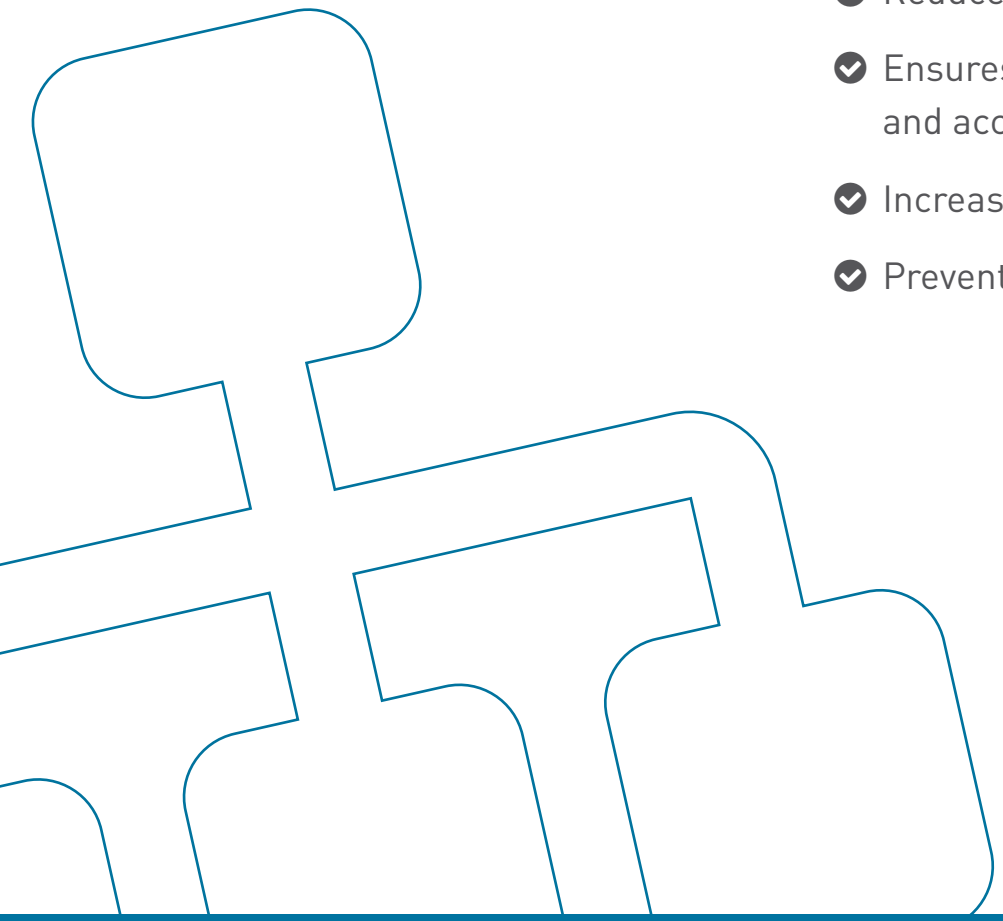
The following sections further explain how to increase test effectiveness under crunching timelines.

02

Advantages

ADVANTAGE OF QUICK AND EFFICIENT TESTING

- Promotes quality
- Achieves on-time delivery
- Reduces the cost of quality (COQ)
- Ensures adequate test coverage, completeness and accuracy
- Increases tester's productivity and motivation
- Prevents delayed identification of bugs



03

Know Your Developers

KNOW YOUR DEVELOPERS

Integration of the development and test teams is one of the main contributors for quick and effective testing. Every quality analyst should understand the purpose of the project, the people working for it and the people using it (users) before testing the application.

This helps during defect/bug interpretation and classification. Developers will be able to accept and appreciate quality analysts based on how they represent themselves and their work to the team.

Every quality analyst should understand the purpose of the project

WAYS TO BUILD RAPPORT BETWEEN THE DEVELOPERS AND TESTERS

ALWAYS PORTRAY THE PURPOSE of a quality analyst as "protection against poor client feedback."

ASK, "What is the most important feature in this project?" to the client and to the developers. This will help in understanding the purpose of the project.

CARRY A POSITIVE OUTLOOK towards the project's progress at all times.

TALK TO THE DEVELOPERS and understand their personalities.

EXPLAIN THE PURPOSE of recorded bugs/features to the developers so that they understand the priority.

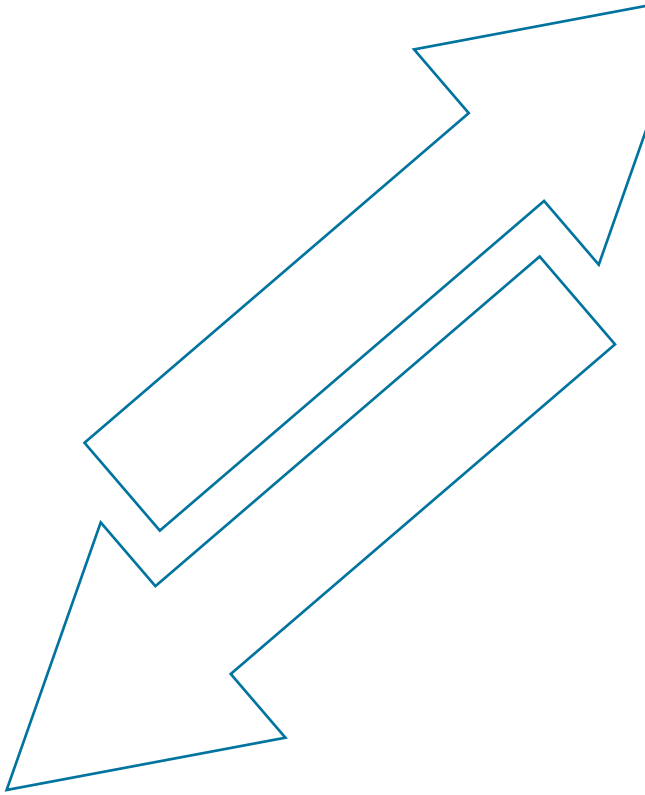
Wherever necessary, **RECONSIDER YOUR DECISION** when a developer feels that a story is not a bug but a feature.

APPRECIATING AND THANKING the developers for their support goes a long way.

WRITE SHORT, simple and clear bugs with test steps so that the developers can work without asking for further explanation.

COMMUNICATE BLOCKS, which are other project commitments/lack of test resources, in advance so that the developers can work around your availability.

Try to **COMPLETE TESTING BEFORE THE ITERATION** end date so that the developers get their velocity points for that week.



04

Tester’s Tongue

TESTER'S TONGUE

It is very important for every tester to master *the art of effective communication and people management.* Conveying bugs never sounds positive but choosing the proper verbiage is very important to set the right context.

A smart quality analyst changes his usage of words depending on the developer's attitude and the criticality of the project. Ultimately the developers should be convinced so that they are willing to fix the bugs immediately.

A FEW EXAMPLES OF HOW TO CHANGE A NEGATIVE TO POSITIVE

WHEN A USER STORY IS REJECTED, say, "I had to restart the story because ...." instead of, "I rejected the story because...."

WHEN MORE THAN TWO USER STORIES ARE REJECTED say, "I accepted most of the stories except ..... because ....."

IF A USER STORY IS NOT CLEAR, ask, "Can you provide test steps for this story?" instead of "I don't know how to test this story."

WHEN ASKED FOR A PROJECT UPDATE say, "I tested all the stories and added a few bugs that require immediate attention. I also added a few UI recommendations that can be discussed whenever you have some time."

WHEN ALL THE STORIES IN AN ITERATION ARE ACCEPTED say, "Good news! I accepted all the stories and there are no issues. Thanks for all the hard work."

WHEN A BUG IS ADDED say, "I added this bug because ....." instead of, "this app/feature is broken."

FINDING BUGS is a tester's gig and not the developer's. When you identify bugs never show your excitement to the developers.

WHEN THERE IS A CRASH say, "I am able to reproduce a crash more than once. I added a bug for it along with test steps. Let me know if you are able to reproduce it," instead of, "The app crashed numerous times and I am not able to test."

WHEN YOU ARE OVERLOADED AND CANNOT TEST say, "Currently I am working on another application but I can test yours later today or definitely by tomorrow. Let me know if you want any specific story to be tested on priority," instead of, "I cannot test today, I am blocked. I will let you know when I can start testing."

A FEW EXAMPLES OF HOW TO CHANGE A NEGATIVE TO POSITIVE (CONT.)

WHEN THE APPLICATION HAS TOO MANY FUNCTIONAL ISSUES say, "The app is designed very well but I found some functional issues and have prioritized the ones that are most important. Can you take a look at it and let me know your thoughts?"

WHEN THE APPLICATION HAS TOO MANY UI ISSUES say, "The app does what it is suppose to do and if we can make a few UI changes then it will be ready for a release. Example: ..."

IN SHORT

ALWAYS MAKE POSITIVE STATEMENTS.

PROVIDE A REASON.

PROVIDE A TEST TIMELINE/DEADLINE.

NEVER SAY "I DON'T KNOW", "I CANNOT TEST", OR "APP IS BROKEN".

THANK THE DEVELOPERS FOR THEIR HARD WORK.

QUOTE EXAMPLES OF SIMILAR APPS ON THE MARKET WHEN MAKING UI RECOMMENDATIONS.

05

Fragmentation Issues

POSITIVE OUTLOOK TOWARDS FRAGMENTATION ISSUES

With so many smartphone competitors in the market, using a single test device will never be sucient to deliver a high quality application. But testing over 200–300 phones is also not possible. In fact, testing more than five phones at a time is a big task with demanding timelines.

FRAGMENTATION ISSUES WILL EXIST FOREVER. Hence testing on every other device in the market is NOT the expectation. A good tester should be able to work around these fragmentation issues and ensure adequate test coverage. Many crashes that we encounter on various phones mostly boil down to low-memory related issues. If we target a capable set of devices, then the main fragmentation problem is only the dierent resolutions and aspect ratios. This is a much simpler problem when compared to all the other functional bugs we identify. We can only plan for minimizing the fragmentation issues instead of removing them, and this should be explicitly discussed with your clients so that they also get the real picture.

A FEW WAYS TO MINIMIZE FRAGMENTATION ISSUES

SELECT THE RIGHT SET of phones
One phone / software version
One phone / manufacturer.

PICK THE MOST POPULAR phones based on usage and the current market penetration.

DO NOT TEST on very old versions or phones.

UNDERSTAND THE DIFFERENCE between the various versions of the phones and the software. This will help you identify all the test cases that can be tested on a single device and be assured that it will work on the other devices.

USER STORIES that involve minor text changes can be tested on a single phone.

GET THE LIST of phones that the client is using for testing.

MAKE SURE that the contract has a list of phones on which the app can run smoothly.

This is a much simpler problem when compared to all the other functional bugs we identify

06

Reduce Testing Time

STEPS TO FOLLOW TO REDUCE TESTING TIME

1. Prepare a test log document before testing. **THIS SAVES TIME DURING TESTING.**
2. **IDENTIFY COMMON TEST CASES** that can be reused across applications.
3. **AVOID** testing too many devices (refer to section 4).
4. **NEVER RELY COMPLETELY ON SIMULATORS/ EMULATORS.** Manual testing on actual devices is always accurate.
5. **ALWAYS TRACK CRASH REPORTS** to avoid reproducing the same bug /crash.
6. **UNDERSTAND** which part of the code is changed and test those screens along with dependencies.
7. **USE FLURRY ANALYTICS** to track events. Understand which part of the application is used extensively and test those screens in detail.
8. Always check the quality of the phones before testing. Make sure there are **NO SOFTWARE/ HARDWARE ISSUES.**
9. **TEST** the application on wifi and run the performance related test cases on 3G. All test cases need not be repeated on 3G.

STEPS TO FOLLOW TO REDUCE TESTING TIME (CONT.)

10 **EDUCATE THE TEAM** to follow a KANBAN approach where small chunks of stories are delivered at regular intervals and the application is frequently tested.

11 **STOP REGRESSION TESTING** if more than two major bugs are identified in the first few test cases. Switch to another project until these bugs are fixed and delivered.

12 **WHILE REGRESSION TESTING**, start with the test cases that failed in the past (mostly performance related test cases). If these test cases fail again, then stop regression testing and report the issues immediately.

13 If you are testing **MORE THAN ONE APPLICATION**, try to switch between them every two hours. You will be able to look at the application with a different perspective and will be able to find more bugs.

14 **LOG AND PRIORITIZE** bugs as you find them. This will help the developers to fix the issues immediately and you will be able to test simultaneously.

15 **ALWAYS REGRESSION TEST THE PRODUCTION BUILD.** Staging environments will never have the same data set as production.

16 **TEST ALL THE DEVICES IN PARALLEL** so that regression testing can be stopped if more than two major bugs are identified in the first few test cases.

17 **CONDUCT UI/UX REVIEWS PRIOR TO TESTING** to capture the design flaws even before regression.

07

Topics to Read

TOPICS TO READ

- KANBAN APPROACH
- AGILE TESTING: A PRACTICAL GUIDE FOR TESTERS AND AGILE TEAMS
- TIPS FOR AGILE TESTING
- AGILE TESTING
- SCRUM

GET STARTED ON BETTER TESTING WITH CHAIONE

Do you need help understanding how to improve your development and testing strategy? Sign up today for a free 30 minute consultation on how you can improve your testing process.

FREE 30 MINUTE CONSULTATION

chaione

Visit **chaione.com** for more awesome content