White Paper

eInfochips
An Arrow Company

The Solutions People

The Essential Guide to Quality Assurance in Digital Transformation Planning

www.einfochips.com | marketing@einfochips.com

CONTENTS

Abstract

The digital transformation journey includes a variety of processes, transactions, interactions, technological changes, evolutions, external and internal factors, and many other parameters. So, it is a complex undertaking.

At the same time, the demand for faster cycle times and service excellence becomes more and more insistent. Therefore, when it comes to assuring quality throughout the digital transformation journey, a wide range of parameters and a set of fresh expectations need to be taken into consideration. When customer experience and full-throttle performance have become the focus, the cost of a miss in quality will have huge ramifications on the product and brand.

In this scenario, companies encounter some common challenges while maintaining and testing the quality of software or hardware, for their products and applications. This white paper will discuss these challenges and offer possible solutions.

Introduction

Almost every business that plans to grow in this competitive market is going digital. Regardless of the size of the business, digital transformation is one of the top priorities for every business leader. The topic of how a business can remain relevant and competitive in this increasingly digital world has become the key focus of panel discussions, keynotes, research papers, and articles.

As digital transformation may look different for every business, it gets difficult to define the concept in a way that is right for each one of them. In simple terms, we can define digital transformation as the integration of digital technology with the aim to create a bridge between the digital and the physical world. It is the convergence of operational technology and information technology. It can be applied to almost all business areas to improve operational efficiency, tap new revenue streams, and to deliver a better experience to customers.

One important question to ask is: Are the products and applications aligned for digital transformation? When a business embarks on the digital transformation journey, it faces challenges at multiple fronts such as software, devices, and system interactions. The key challenge is to achieve transformation while maintaining product/solution quality and interoperability of connected systems while also delivering a delightful user experience.

It is not uncommon for large organizations to spend tons of money as a cost of excess rework resulting from unexpected changes in product functionality or design. Quality assurance (QA) can play a vital role in overcoming such challenges. Automation of test procedures can further reduce the frequency and the number of human errors, improve the test coverage across the systems, and reduce the time-to-market for the products.

During the product growth phase, businesses are more focused on reducing the cost of improving product quality and associated rework time – both of which can be achieved with more effective testing. When the product has matured, the business focus shifts to higher productivity and rapid innovation to stay competitive in the market.

Let's discuss how quality assurance in different areas can contribute to the goal of digital transformation:

1. DevOps Testing

DevOps is becoming an integral part of every organization's strategy as it provides higher speed, frequency, and reliability while engineering products, applications, and solutions. The emphasis is more on business agility and rapid release cycles with continuous integration and continuous deployment.

DevOps helps in enabling effective execution, eliminating repetitive tasks, and automating manual processes to improve the productivity of an organization.

Let's discuss what the DevOps end-to-end implementation cycle entails and explore the role that testing plays in it:

Continuous Integration

This is a process of integrating multiple product development codes (device, web, and mobile) to a central location and validating each one through customized build and test automation.

Continuous integration automates the environment setup and triggers automated code every time the code check-in takes place. It also generates and publishes reports after every check-in and sends notifications in case of success or failure. Various tools like Maven, Ant, Gradle, Jenkins, and Bamboo are available to execute this process.

Continuous Testing

As a part of this process, virtualization/simulation of test environment takes place to enable continuous testing. Performance testing and automation execution are achieved with network virtualization and simulation of physical devices.

Parallel remote execution takes place to provide risk-based feedback as rapidly as possible; this is achieved by tracing test metrics, test cycle time, and MTTR (Mean Time to Resolve). Tools like Selenium, Appium, Python Robot Framework, and TestNG are used to perform continuous testing.

Continuous Deployment

This process helps to manage rollback deployment in a live environment. Auto build deployments on the device and the server allow automated alerts on failure scenarios and performance issues.

In this process, automation is achieved through application-release automation (ARA), which allows the development, test, and production teams to use the same process without the need for one-off deploy scripts to support continuous delivery and deployment requirements.

Tools like Chef, Puppet, Ansible, and Docker can be used to achieve continuous deployment.

Continuous Monitoring

This process is implemented with application performance management (APM) solutions that allow application and infrastructure monitoring. It helps in theft alert and incident management through proactive monitoring and performance analytics.

Microservices help in decomposing the application into smaller, independently running parts. By implementing continuous monitoring, you can get better insights into monitoring metrics. You can use continuous monitoring tools like CloudMonix, NewRelic, and Zabbix for this process.

Challenges in DevOps Testing

- It is difficult to integrate all the complex and fragmented pipelines such as firmware, server-side code, mobile/desktop apps, etc., to ensure quality end-to-end testing.
- Devising and implementing a testing solution for legacy devices such as sensors, cameras, and others is challenging. There are requirements for managing, updating and maintaining existing devices; this increases discrepancies and complexities in DevOps when legacy systems come in the picture.
- Virtualizing multiple connected devices along with the server infrastructure is difficult.
- It is complex to reproduce and maintain an effective production environment when product customization and market segment variations need to be considered.
- A unified release is a challenging task as different components of solutions like firmware, web/mobile/PC app, etc., have a different release cadence.

Benefits of DevOps Testing

- Businesses achieve cost reduction, manual error reduction, and duplicate effort reduction with end-to-end delivery.
- The software development cycle is shortened due to continuous automation. This results in faster and more frequent delivery of features, accelerating the time-to-market.
- Savings on manual cost by developing a more robust and stable process with frequent backups and rollovers through automation.
- Increase in overall development speed due to early detection and documentation of errors through continuous monitoring.
- Proper resource optimization by splitting test and build processes over different machines through continuous integration.
- Expenses saved by shifting from capital expenditure to operating expenses via virtualization (paying only for what is used).

2. Cloud Testing

The need for cloud testing is increasing as more and more organizations have started migrating to cloud solutions like public cloud, private cloud or hybrid cloud. As a result, it becomes necessary to verify and validate some specific cloud functions like performance, scalability, and redundancy.

Cloud testing is the process of testing the performance and reliability of web applications deployed in the cloud environment. Simulated real-world web-traffic is used to test web applications deployed in the cloud environment.

For cloud testing, it is essential to meet functional as well as non-functional requirements of the cloud. These tests are performed to make sure that the business meets the desired requirement, and the cloud offerings efficiently provide the services the user is paying for.

Let's discuss what is included in cloud testing:

Performance Testing

Response time verification is critical to ensure that everything is working fine when there is a lot of workload on the cloud application. Hence, performance testing is done to test the performance of the application during peak load times.

Security Testing

It is essential to make sure that all sensitive user information is protected and is accessible to authorized users only. Data must be encrypted when in use and deleted when no longer in use to provide data security. It is also required to regularly measure the ability of the cloud to withstand data breaches and equip the system with the latest defense security controls.

Interoperability and Compatibility Testing

The application should be tested for interoperability before deploying it on the cloud. It must also have the flexibility to work seamlessly across different platforms, in case it is moved to another cloud infrastructure. Hence, compatibility testing also forms an important part of cloud testing.

Multi-Tenancy Testing

Testing must be implemented to ensure adequate security and access control for all the users connected to a single instance of a multi-tenant architecture.

Disaster Recovery Testing

Any application failure must be indicated in case of network outages, extreme-load breakdowns, and system failures. It is necessary to measure how fast the failure is observed and if any data loss occurs during that period. Here, disaster recovery testing is helpful.

System Verification and Acceptance Testing

System verification and acceptance testing ensures that the system behavior is as expected and all the components are working fine together. It also ensures that the users' expectations from the cloud solutions are met.

Load and Stress Testing

Load and stress testing help in the identification of system failures under increasing load, defects responsible for environment failure, changes in the system over time under a certain load, and performance optimization with the increase or decrease in application load.

On-premise application testing on cloud

The cloud is often leveraged as an Infrastructure as a Service (IaaS), which is availed via private or public cloud models to overcome the limitations of the existing infrastructure. Testing needs to be done at the functional, performance, and security levels for an on-premise application on cloud. Benchmarking exercise is also required to check application performance in an on-premise environment with the cloud environment.

Cloud migration (Application/Database/Server/OS) testing

An application needs to be tested while analyzing its stability requirements; assessing compatibility when moving from legacy to new infrastructure; and also when adopting new technologies. A database needs to be tested through various mapping fields, validation points, pre- and post-migration accuracy techniques,

and more. Also, server and OS testing are needed to check the compatibility, usability, data liability, and dependencies of the application.

Challenges in Cloud Testing

- It is a challenge to ensure that live upgrades do not impact the existing connected users while simulating live upgrade testing.
- The risk of data theft is always present in a multi-tenant cloud environment. Maintaining user privacy protection, security standards on the cloud, and security of applications running on the cloud is a challenge.
- Changes in servers, storage or networking pose a major challenge – in case of change in cloud service provider.
- Different cloud providers use different database schemas. This makes migrating data from one cloud provider to another a huge problem as it is difficult to understand data fields, relationships, and SaaS application mapping for different cloud providers.
- The application in a private cloud could be shared by multiple users. This delays the response in case of insufficient bandwidth allocation to test and check the performance of any specific application.

Benefits of Cloud Testing

- Transferring to cloud helps to save costs associated with unutilized server resources and license purchases/renewals.
- Users can get unlimited access to the cloud environment for testing as long as the internet connectivity is available.
- Testing becomes more efficient and effective with resource virtualization and customization.
- Due to seamless software upgrades, zero downtime, and quick set up & tool deployment, this testing becomes quicker than the traditional methods.
- Due to the immediate availability of all the tools, testing becomes agile and testers can perform testing under larger scenarios with optimum resource availability.

3. IoT Testing

We are aware of the upsides the IoT (Internet of Things) can bring to businesses; however, its potential downsides also need our consideration. When we talk about implementing an IoT project, security concerns inevitably creep into the conversation.

> *According to a study by Hewlett-Packard, 70% of IoT devices are vulnerable to attack.*

As the number of IoT devices is expected to grow significantly in the next decade, testing these devices — ranging from a smart refrigerator that can automatically order milk, to self-driving cars that can identify obstacles on the road – will be the most challenging part.

IoT testing is performed to ensure that IoT devices function as intended and are capable of communicating effectively with other devices within the network. Testing these devices in a controlled lab environment will be far different from testing them in the actual environment where they will be deployed.

Here are five common types of tests which can be used for IoT devices:

Interoperability Testing

The IoT ecosystem comprises plenty of devices from different manufacturers, each of them working on different configurations and varying set of standards. Interoperability testing is used to make sure that all these devices from different vendors work together as defined by the standards.

API Testing

On top of business data, users' data can also be at risk while using APIs. So, API testing is really important. Thankfully, it is really easy to execute. API testing can be performed by testing all the endpoints, regardless of where they are hosted.

Reliability Testing

A bunch of sensors and actuators are used to get real-time information about the IoT devices. Reliability testing will verify the proper functioning of all the IoT components, which collectively make a reliable IoT environment.

Security Testing

In an IoT network, multiple users have access to a great amount of data. Security testing is a mandatory step to verify user identity and access control via authentication. Data privacy controls must be a part of security testing.

Performance Testing

The efficiency of the end-to-end IoT ecosystem needs to be checked beforehand, which makes performance testing an integral phase of testing. Performance testing is important to ensure the expected outcome is delivered by a company's product or offerings while creating an IoT ecosystem.

Challenges of IoT Testing

- The operation of every IoT device in the network depends on various software and hardware components. With version upgradation in the hardware and software, it gets difficult to test all the permutations and combinations of all the available versions.
- The performance of IoT devices largely depends on the network configuration. So, if there are inconsistencies in the internet connection or encumbrances in the channels, testing needs to be done for all the network conditions.
- IoT is a combination of different types of devices that belong to different platforms and are produced by different manufacturers. So it is really difficult to evaluate the types of sensors required, formats of data storage, and application behavior across different platforms for each individual device.
- The modern Software Development Life Cycle (SDLC) requires the monitoring of end-to-end IoT ecosystem (starting from the sensors that generate data, to the cloud that stores the data) to power continuous deployment and delivery pipelines.
- Security is a major concern for IoT projects as every device is connected to sensors that collect device information, which is then analyzed and stored in the cloud. All of these devices are connected to the Internet and are constantly communicating with other devices in the ecosystem.
- When using APIs, the risk associated is much higher than a bug in the application's user interface.
- Monitoring both internal device communication as well as device communication with the network is challenging.
- Limited resources in terms of processing power, memory, battery life, and bandwidth.

IoT Testing Best Practices

- A real-time operating system (RTOS) is essential to deliver connectivity, modularity, security, reliability, and scalability, all of which are important for an IoT system.
- Effective test cases can be prepared using gray box testing to know the architecture, operating system, third-party hardware, and limitations of hardware devices.
- IoT testing should be automated to increase testing productivity, remove hardware resource bottlenecks, scale to current and future needs, and to improve performance and security.

4. Cognitive QA

Cognitive computing is leading the drive towards innovation. Conversational interface or virtual assistant applications such as Siri, smart shopping stores like Amazon Go, Facebook's face recognition feature, and even Google's autonomous cars are designed to think and behave just like humans. All of these systems work on algorithms that thrive on data. The more data available to train or test the system, the smarter it becomes, and in turn, the more accurate are the results.

Advances in artificial intelligence and machine learning have helped to build prediction in the QA process. This requires changes in the organization's culture and approach to testing a digital experience for the customer. Cognitive QA is about applying machine learning, predictive, and prescriptive techniques for testing. A self-learning and self-adaptive environment is the key to cognitive QA.

When we talk about Artificial Intelligence and Testing, there are two scenarios we need to discuss:

- Testing with AI
- Testing of AI

Testing with AI

The traditional approach of software testing had longer delivery cycles, which was shortened by the adoption of Agile and DevOps techniques, and through continuous testing. Yet, continuous testing will not be enough if we look at the future of testing. We will need machines that can mimic human intelligence and can apply predictive analytics and machine learning algorithms to accelerate the software delivery cycle, which can be achieved using cognitive QA.

Artificial intelligence for testing software and applications can help in making smarter decisions based on factual data, user feedback, and actual usage patterns. This makes is possible to deliver better results with improved speed and cost.

While testing with AI, the ground truth will be the data that you use to train and test the AI system. Throughout the learning phase, this dataset will be considered as the absolute truth or the primary point of reference. Its layout and structure may vary according to the type of the system that is being created; however, it will always abide by a number of set rules.

Testing of AI

To determine whether an AI system is capable of delivering the expected results, it needs to undergo several steps of testing. So, if you are dealing with AI in the form of physical robots or devices, their electrical and mechanical aspects need to be tested before moving further.

When using machine learning algorithms to process data, you cannot expect a single correct output. Instead, you need to define a list of outputs that can be labeled as correct. So, during the second phase, you need to check if the output provided by the AI under analysis is providing the expected results.

The next step will be to check for the business and the social impact of the AI system – analyze if they meet the primary expectations that were set before beginning the project.

Challenges in Cognitive QA

- High level of dependency on the data: one needs to rely on the transparency, quality, and size of data in order to derive the metrics that enable the testing operations to predict the quality of applications based on the data.
- Even after gaining business insights and reports, implementing those insights in the real world becomes a challenge.
- To successfully test an AI system, the testing team will need to develop a new set of skills. They will need to build knowledge of machine learning, big data, mathematics, and statistics, plus have experience of working with all the relevant testing tools.
- When working with AI and ML, the input data needs to be strictly monitored. Any changes in, or manipulation of, the input data will result in a changed output and lead to hampered security.
- Certain legal aspects also enter the picture when it comes to collecting and storing the data in a central database. It is critical to ensure that the data collection, storage, and processing techniques comply with the relevant data protection and privacy regulations.

Despite the above-mentioned challenges, when cognitive QA is put into implementation, it opens up a wide range of possibilities across different industries.

Use Cases for Cognitive QA

Using cognitive algorithms during the different phases of the application lifecycle will deliver enormous savings in terms of cost, quality, and time.

Let's have a look at the top use cases for cognitive QA:

- An AI-based system can completely automate test designs, suggest better implementation strategies, increase test coverage, and develop the capability to predict the modules that are more likely to have defects.
- An AI-powered system can be fed with production data to identify frequently used processes and functions. So, whenever a new requirement comes up, machine learning techniques can be leveraged to easily identify the changes required in each of the modules to get started with the production. This will significantly decrease the execution efforts and time required for regression testing.
- Cognitive QA can be applied to check if the system is capable of accurately identifying and understanding human speech using natural language processing (NLP) and can respond to human commands in a human-like manner.
- AI systems with machine learning capabilities can be integrated into products like Amazon Echo to learn and remember its users' preferences and use this knowledge to enhance future interactions.

5. Chatbot Testing

Conversational interfaces or chatbots are going to change the way customers interact with brands and businesses. Many industry leaders believe that conversational interfaces will replace websites and applications as they use the most natural way of human communication — dialogues.

> *According to HubSpot, 55% of consumers are interested in interacting with a business using messaging apps to solve a problem.*

In today's technology landscape, there is a lot of hype surrounding chatbots and their applications across various industry verticals. Creating a chatbot that can still be relevant and useful after the hype fades is a challenge. To make chatbots user-friendly, it is necessary to ensure that it performs its tasks accurately and seamlessly, while also being interactive.

How to create a chatbot that stays relevant over time? Building one that functions as intended and delivers a good user experience is the key.

Just like any other software or application, a chatbot should also be tested before deployment. However, chatbot testing might require a different approach and procedure. Let's have a look at some of the chatbot testing methods:

Functional Testing

This is the first and foremost testing method that verifies if a chatbot can efficiently perform the defined functions. This type of testing can be performed using common mechanisms such as equivalence partitioning and boundary value analysis. This testing is to ensure that the chatbot answers with a valid response to a valid request.

Conversation Testing

No user would like to chat with a bot that does not have a pleasing personality or proper conversational flow. Conversation flow testing ensures that the bot is able to maintain a proper conversational flow and is able to take actions that are expected from a human assistant with the same responsibilities.

Context Testing

Understanding users' messages is the key to process their requests. Natural language processing is used to understand the context of messages and it also allows chatbots to generate human-like replies. Context is what continues the conversation, so it is important to test the bot's understanding of different inputs, including small talk and slang, provided via text or voice. A chatbot should be able to provide satisfactory responses to the domain-specific questions and maintain a high correct answer ratio.

Limit Testing

Context and functions are tested when the user communicates as expected. However, when the user moves away from the regular expressions and stops sending meaningful messages, the limitation of the chatbot should also be defined. This phase of testing validates how the chatbot will respond to invalid requests.

Intelligence Testing

This is the phase of testing that helps to determine that chatbot's ability remember the conversation and the context of the previous interaction with a specific user. An intelligent chatbot should be able to automatically understand the context in such continuous conversations.

Challenges of Chatbot Testing

- The chatbot's ability to understand natural language plays a major role for bots that run on voice commands.
- Chatbots may get confused when the user interaction gets a little complicated or when the user is coming up with multiple conversation topics at the same time.
- When working on voice commands, the language or tone of a particular user might also pose a challenge for the voice recognition system.
- Users have no barriers while using chatbots and are free to ask anything and everything, so an unexpected question or request could confuse the chatbot.

Benefits of Chatbot Testing

- Develops chatbots that can competently perform their domain functions.
- Helps chatbots in understanding the context of conversations with its users.
- Develops a better ability to manage errors and respond to irregular conversations.
- Creates a chatbot personality that is interesting enough for the target users.
- Gives chatbots the ability to respond to requests in a human-like manner.

Bonus content: Voicebot Testing

As the voicebot ecosystem continues to grow at a rapid pace, the demand for testing voice apps and products will become more and more insistent. Smart speakers like Amazon Echo, Google Home and Sonos One are witnessing huge sales – and these are just a tip of the iceberg in terms of voice-activated systems.

Over the coming years, we can expect to see more and more voice-controlled appliances capture the imagination of users over the next few years. Needless to say, the testing of voicebots and voice-activated systems comes with its own set of challenges.

Let's take a close look at the development and testing cycle for Alexa skills. This will help us identify the common challenges on the path of voicebot testing, plus offer guidance on how to overcome them.

Alexa Testing

There are many opportunities available to develop custom Alexa skills, which can help deliver an innovative user experience. While developing an Alexa skill, a developer must focus on policy, security, functionality, and voice interface for building a high-quality experience for users.

There are certain consequences a developer may face while developing an Alexa skill. Here are some steps and best practices to avoid such consequences:

- For an accurate Alexa speech recognition, provide three example phrases as a part of skill submission.
- Select an appropriate invocation name to engage a customer properly with your skill.
- Provide proper documents like a signed confirmation letter or send a license via email to confirm IP ownership for avoiding IP infringement issues.
- Proper session management needs to be done to understand whether the conversation with consumer needs to stay open or closed.
- Provide helpful instructions regarding the core functionality of skill when a user asks for help.

To collect Works with Alexa badge, you need to go through the certification process as below:

- Get the submission form and submit the test scope with the product types, details on how they connect with Alexa, the total number of devices, and service end-point location (geography-wise).
- Submit your products for testing by accepting the terms and conditions and paying the testing fees.
- Go through the device testing process and collect the certification.

After completing this certification process, the 'Works with Alexa' badge will be applied to your products on Amazon.com, which can be used on packaging and other marketing materials.

Now, let's jump into the process of testing Alexa skills using test automation below:

Manual Testing

This testing helps to identify any obvious issues that arise while simultaneously testing the code, interaction model, and user experience. There are various tools available for this type of testing. You can use any Echo device for testing using voice, or utilize virtual solutions like echosim.io or Reverb.ai.

In case you want to test without speaking a word, you can use the Alexa Test Simulator or Bespoken Utter and Speak commands that provide an interface to interact via text, speech, and visual displays.

Unit (Integration) Testing

This testing helps to verify and ensure that each object and a major portion of skill code functionality work as intended. Unit test frameworks are used to run unit tests; some popular frameworks are Mocha, Jasmine, QUnit, and Jest. Assertion frameworks are used to compare actual and expected results of tests; some popular frameworks include Chai, Sinon, and Cucumber.

With code coverage, it is easy to find problem areas in the skill code. AWS Lambda is a skill handler used for code coverage in Alexa skill code testing. Continuous integration is used to collate source code, unit tests, and code coverage by alerting the developer in case of any issues. This helps to improve the code quality.

End-To-End (Deployment) Testing

This is the process of testing a skill as a whole: checking a particular phrase, skill behavior, external

components like Amazon S3 and Amazon DynamoDB, and detecting any issues. Several virtual devices are available to perform full cycle Alexa testing without speaking.

Continuous deployment is used to check the latest code version available for testing. It helps to update the interaction model for the skill using the ASK CLI tool through which it is easy to implement automation for Alexa skills. It is also useful in running sanity tests and regression tests before the deployments are approved.

Monitoring and Testing

This testing helps to make sure that the Alexa skill is working properly at the development stage and after deployment as well. Cloud monitoring tools like AppDynamics, CloudMonix, Slack, and PagerDuty can be leveraged for continuous testing. Amazon CloudWatch is also helpful in triggering an alarm and notification in case of emergency.

By enabling your device with the 'Works with Alexa' certification, you deliver an amazing experience to your customer: they can use voice commands for the Alexa smart home device, access multiple Alexa skills, stream music, listen to books and so much more.

Conclusion

After brainstorming product/solution/application ideas and creating a working prototype that reflects those ideas, testing is a critical function that ensures your products or solutions are able to satisfy the expectations of your end users. The business goal of building a digital enterprise can only be achieved using smart QA, which ensures product reliability and elevates the user experience.

In a demanding market where one bad experience can severely damage the reputation of your product or application, quality assurance takes the front seat. Make sure that your quality assurance process helps you build and release faultless solutions that offer the exact answer to the user concerns while being efficient and easy to use.

About eInfochips

eInfochips, an Arrow company, is a leading global provider of product engineering and semiconductor design services. With over 500+ products developed and 40M deployments in 140 countries, eInfochips continues to fuel technological innovations in multiple verticals. The company's service offerings include digital transformation and connected IoT solutions across various cloud platforms, including AWS and Azure.

Along with Arrow's $27B in revenues, 19,000 employees, and 345 locations serving over 80 countries, eInfochips is primed to accelerate connected products innovation for 150,000+ global clients. eInfochips acts as a catalyst to Arrow's Sensor-to-Sunset initiative and offers complete edge-to-cloud capabilities for its clients through Arrow Connect.