Different types of tests that are used to analyze the EHR)

Name: Ali Moustafa Ahmed Mohamed

ID: 20208163

Group: 4

Introduction

Ensuring that EHRs are safe and effective is an important and increasingly visible challenge, The quality and safety benefits of EHRs have been long understood, but unintended consequences, malfunctions, and safety issues have also been reported, Recently, the Office of the National Coordinator for Health Information Technology (ONC) released a report focused on safety issues in EHRs, and also commissioned the development of a series of proactive EHR risk assessment tools, called the Safety Assurance Factors for EHR Resilience (SAFER) guides. These SAFER guides identify pre-implementation and ongoing testing of EHRs as a key safety practice, and the importance of testing software is, of course, already widely emphasized.

In our research, we have undertaken a number of cross-organization evaluations of EHRs as implemented at sites across the country. In each of these projects, we have worked with clinicians, and IT personnel to conduct evaluations of their EHRs. In addition to production environments, most sites also have test environments that are designed to mirror the production EHR. Whenever possible, we requested that evaluations be conducted in the "live" production EHR, the same one in use by clinicians, using test patients. In most cases, the sites were able to accommodate our request to test in production, but a minority of sites reported a policy against testing in the production EHR or did not have any test patients configured for testing in production. In other cases, there were significant limitations that prevented reasonable testing in the production environment, such as no test patients admitted to hospital beds (preventing any inpatient testing) or no ability to create laboratory results on test patients in the production

environment. In these cases, we fell back to testing in a test or support system environment rather than the production environment.

Impact of EHRs on Care

Our world has been radically transformed by digital technology – smart phones, tablets, and web-enabled devices have transformed our daily lives and the way we communicate. Medicine is an information-rich enterprise.

A greater and more seamless flow of information within a digital health care infrastructure, created by electronic health records (EHRs), encompasses and leverages digital progress and can transform the way care is delivered and compensated.

Patient access is important as it has shown that when patients have access to their personal health records, they will take an active role in managing their healthcare on a day-to-day basis. Today's patient portals provide lab results, drug information, and historical data while also extending a convenient way to schedule appointments, refill prescriptions, and communicate electronically

With EHRs, information is available whenever and wherever it is needed.

.

- ✓ **Improved Patient Care**
- ✓ **Increase Patient Participation**
- ✓ **Improved Care Coordination**
- ✓ **Improved Diagnostics & Patient Outcomes**
- ✓ **Practice Efficiencies and Cost Savings**

Why Do EHR systems Need to be Tested?

The Electronic Health Record (EHR) system is an important cog in the wheel of healthcare management. It needs to be safe and quickly accessible (if possible, remotely) to ensure better monitoring and management of patients' health. Since the record system may have interfaces with other hardware and software suites, it should be tested in the production environment for better efficacy. However, in many cases, the healthcare software testing specialists limit testing to a test environment, which should be avoided. This is because a production environment can have myriad variables and situations that cannot be mimicked in a test environment.

If the EHR systems are not tested for such variables or situations, there might be gaps left unchecked, which might come to haunt users later. Hence, effective EHR system testing can only be ensured if the testing process integrates the production environment comprising patients. The healthcare software testing should incorporate elements like the release of upgrades, downstream personnel, documentation, and test patients.

The stakeholders understand the benefits of EHR in terms of quality and safety. However, there have been reports of the system malfunctioning or generating erroneous data leading to healthcare management compromise. Also, their usage by the end-users namely, doctors, nurses, and other healthcare staff leaves a lot to be desired. It has been noticed that usability issues can render the very use of such systems infructuous. This calls for **testing EHR systems** where usability issues should be flagged and fixed and pre-empt patients from coming in harm's way. Test experts usually rely on creating a virtual test environment mirroring the real environment wherein they conduct a series of healthcare application testing**.** It is not always the case that testing in a production environment shall yield the best results. For example, without getting access to real patients or creating laboratory results on such patients, it will not make the healthcare QA testing effective. But one cannot take away the advantages of testing in a production environment. Let us find out the reasons.

WHAT IS TESTING?

Software Testing Type is a categorization of various testing activities into groups, each of which has a specific test objective, test plan, and test outputs. Validating the Application Under Test (AUT) for the specified Test Objective is the aim of having a testing type.

For instance, accessibility testing aims to confirm that the AUT is usable by individuals with disabilities. So, if your software solution needs to be accessible to people with disabilities, you assess it against the accessibility test.

Functionality

EHRs feature multiple workflows, from patient admissions and insurance verifications to obtaining laboratory results and issuing medication prescriptions. Also, the workflows may vary among healthcare professionals and among different healthcare facilities. During functional testing, a test engineer needs to verify that each workflow complies with the requirements specification.

To achieve optimal test coverage without compromising on testing quality, functional testing of EHR software can be performed based on the functional importance of workflows and the business risk associated with their not functioning as intended. For instance, as *medication order prescription* is critical in both aspects, it is classified as high-priority and tested first, with multiple test cases and across different environments. In its turn, *medication order search* is of less criticality, hence, the workflow is assigned a medium priority and tested once and only in the testing environment.

What is the need of Testing?

Software flaws may be expensive or even harmful, thus testing is crucial. History is replete with instances where software defects led to financial and human loss.

- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- Nissan cars recalled over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
- Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point, the store served coffee for free as they were unable to process the transaction.
- Some of Amazon's third-party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
- Vulnerability in Windows 10. This bug enables users to escape from security sandboxes through a flaw in the

  win32k

4 Reasons Why Software Testing is Important

1. Easy while adding new features

The more interconnected and older the code, the more difficult it is to change. Tests counteract this calcification tendency by allowing developers to confidently add new features. As a new developer, changing older parts of your codebase can be terrifying, but with tests, you'll at least know if you've broken anything important. This helps in making your software stand ahead in the market, and beat the competition.

2. Determining the performance of the software

If you find software or application that has low or reduced performance, you will find that it brings your reputation down in the market.

Users are not going to trust any people. There are chances that the reputation of your organization is going to suffer.

3. Satisfaction of the customer

The primary objective of the owner of the products is offering the best satisfaction of the customers.

The reasons why it is necessary to opt for **software testing** is due to the fact that it offers the prerequisite and perfect user experience.
As you opt for the best project in the saturated project, you will be capable of earning the reputation of reliable clients.

Thus, you are going to reap long-term benefits by opting for **software testing**. Earning the trust of the client is certainly not an easy task, primarily in case the product is found to be functioning and glitching every time or the other.

4. Security

It is another crucial point why **software testing** should not be taken into consideration.

It is considered to be the most **vulnerable** and sensitive part. There are a bunch of situations in which the information and details of the users are stolen and they are used for the benefits.

It is considered to be the reason why people look for the well tested and reliable products.

TYPES OF TESTING?

- Integration tests.
- Functional tests.
- End-to-end tests.
- Acceptance testing.
- Performance testing.
- Smoke testing.
- **White Box Testing.**
- **black-box testing.**
- Unit Testing
- System Testing
- Top-down Incremental Integration Testing
- **Non-functional testing**
- Usability Testing
- Compatibility Testing
- Load Testing
- Stress Testing

1. Unit Testing

Unit testing is the first level of functional testing in order to test any software. In this, the test engineer will test the module of an application independently or test all the module functionality is called **unit testing**.

The primary objective of executing the unit testing is to confirm the unit components with their performance. Here, a unit is defined as a single testable function of a software or an application. And it is verified throughout the specified application development phase.

For example, there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

2. Integration Testing

Once we are successfully implementing the unit testing, we will go integration testing. It is the second level of functional testing, where we test the data flow between dependent modules or interface between two features is called **integration testing**.

The purpose of executing the integration testing is to test the statement's accuracy between each module.

For example, a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

Types of Integration Testing

Integration testing is also further divided into the following parts:

- **Incremental Testing**
- **Non-Incremental Testing**

3. Top-down Incremental Integration Testing

In this approach, we will add the modules step by step or incrementally and test the data flow between them. We have to ensure that the modules we are adding are the **child of the earlier ones**.

4. System Testing

Whenever we are done with the unit and integration testing, we can proceed with the system testing.

In system testing, the test environment is parallel to the production environment. It is also known as **end-to-end** testing.

In this type of testing, we will undergo each attribute of the software and test if the end feature works according to the business requirement. And analysis the software product as a complete system.

5. Non-function Testing

The next part of black-box testing is **non-functional testing**. It provides detailed information on software product performance and used technologies.

Non-functional testing will help us minimize the risk of production and related costs of the software.

Non-functional testing is a combination of **performance, load, stress, usability and, compatibility testing**.

6. Performance Testing

In performance testing, the test engineer will test the working of an application by applying some load.

In this type of non-functional testing, the test engineer will only focus on several aspects, such as **Response time, Load, scalability, and Stability** of the software or an application.

7. Load testing

While executing the performance testing, we will apply some load on the particular application to check the application's performance, known as **load testing**. Here, the load could be less than or equal to the desired load.

It will help us to detect the highest operating volume of the software and bottlenecks.

8.Stress testing

It is used to analyze the user-friendliness and robustness of the software beyond the common functional limits.

Primarily, stress testing is used for critical software, but it can also be used for all types of software applications.

9. Usability Testing

Another type of **non-functional testing** is **usability testing**. In usability testing, we will analyze the user-friendliness of an application and detect the bugs in the software's end-user interface.

Here, the term **user-friendliness** defines the following aspects of an application:

- The application should be easy to understand, which means that all the features must be visible to end-users.
- The application's look and feel should be good that means the application should be pleasant looking and make a feel to the end-user to use it.

10. Compatibility Testing

In compatibility testing, we will check the functionality of an application in specific hardware and software environments. Once the application is functionally stable then only, we go for **compatibility testing**.

Here, **software** means we can test the application on the different operating systems and other browsers, and **hardware** means we can test the application on different sizes.

11.Smoke testing

Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level.

Whenever a new build is provided by the development team, then the Software Testing team validates the build and ensures that no major issue exists. The testing team will ensure that the build is stable, and a detailed level of testing will be carried out further.

For example, tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the

application. Smoke testing for this website verifies that all these functionalities are working fine before doing any in-depth testing.

12. Black box testing

Black box testing is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects

13. End to End Testing

It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

For example, a tester is testing a pet insurance website. End to End testing involves testing of buying an insurance policy, LPM, tag, adding another pet, updating credit card information on users' accounts, updating user address information, receiving order confirmation emails and policy documents.

Conclusion

In summary, we believe that testing in the production environment is essential for ensuring the safety of EHRs. The recommended practices, taken together, will increase the fidelity of EHR testing while ensuring safety and preventing harm during the testing process. Organizations with policies that prevent testing in the

production environment should carefully reconsider their approaches.

The above-mentioned Software Testing Types are just a part of testing.

However, there is still a list of over 100+ types of testing, but we do not use all testing types in all types of projects. Hence, we have covered some common Types of Software Testing which are mostly used in the testing life cycle.

REFERENCES

1. https://www.javatpoint.com/types-of-software-testing

2. Institute of Medicine. *Health IT and Patient Safety: Building Safer Systems for Better Care*. Washington, DC: National Academies Press; 2012. [Google Scholar]

3. RTI International. *Health IT Safety Center Roadmap*. Washington, DC: Office of the National Coordinator for Health Information Technology; 2015. http://www.healthitsafety.org/uploads/4/3/6/4/43647387/roadmap.pdf. Accessed March 28, 2016. [Google Scholar]

4. Office of the National Coordinator for Health Information Technology. *Health Information Technology Patient Safety Action and Surveillance Plan*. 2013. https://www.healthit.gov/sites/default/files/safety_plan_master.pdf. Accessed March 28, 2016. [Google Scholar]

5. Office of the National Coordinator for Health Information Technology. *Recent Evidence that Health IT Improves Patient Safety* (Issue Brief No. HHS P23337047T). Washington, DC: Banger A, Graber ML, RTI International; 2015. [Google Scholar]

6. Buntin MB, Burke MF, Hoaglin MC, et al. The benefits of health information technology: a review of the recent literature shows predominantly positive results. *Health Aff (Millwood).* 2011;30(3):464–471. [PubMed] [Google Scholar]https://www.softwaretestinghelp.com/types-of-software-testing/

https://www.guru99.com/software-testing-introduction-importance.htm