ISSN 2347 - 3983
Volume 8. No. 10, October 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter1068102020.pdf
https://doi.org/10.30534/ijeter/2020/1068102020

# Evolution of the Testing Strategies using Advanced Automation and Artificial Intelligence in Software Development

**Satyanandaram Nandigam[1], Smitha Chowdary Ch[2]**
[1]Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation , India, satyanandaramn@gmail.com
[2]Associate Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, India, smithacsc@gmail.com

## ABSTRACT

The main aim of artificial intelligence in software engineering is to help teams to develop and test their code more efficiently and effectively, to create higher quality software at speed. Software Engineering is having various phases to be followed. The phases are Plan, code, build, test, release, deploy, operate and monitor. By implementing the automated testing patterns, the testing automation will be done for service-based AI systems. Software testing is the process of evaluating the developed system to calculate the quality of the final product or software, but the software testing process is very costly and takes a lot of time through the software development life cycle and manual software testing becomes more and more difficult. Therefore, we just need to decrease the testing time. Recently, smart testing is a major factor in reducing the testing effort using machine learning techniques. This paper aims to compare the main features of different software testing techniques.

**Key words:** Artificial Intelligence, Automation, Autonomous Testing, Continuous testing, DevOps, integrated tools, Real-time, Source control, Test Driven.

## 1. INTRODUCTION

Artificial intelligence is the study of the fundamental limits of computing. AI helps us to discover a new class of computing problems [1]. If any new class of problem is identified, then it acquires a name and the engineers understand the problem. The tools will be implemented by the engineers for addressing the problem. Many AI applications are extremely difficult and time-consuming to build and it requires vigilant analysis, modeling, system design, engineering, test, and evaluation[2]. The skills in software engineering help to build successful AI systems. Software engineering derives efficient access to develop applications. It is providing an organized set of past experiences [3]. These are arranged as guidelines and methodologies. Generally, to implement a smaller system requires programming. To implement a bigger system like artificial intelligence systems should follow the software engineering technics. By incorporating the technics one can develop bigger software products and achieve quality systems [4]. The purposes of AI systems are to automate the services of systems to end-users. The advanced software engineering culture like DevOps will provide the ability of continuous communication between the development teams and operations teams [5]. The development team will build the system and deploy the system over the cloud. Here I am going to analyze an automated testing pattern for service-focused AI systems

## 2. LITERATURE REVIEW

Software testing is a process to check whether the actual results match the expected results and to make sure that the developed software is defect-free [6]. Software testing also helps to identify errors and find missing requirements to the actual requirements. It can be done in two ways in manually and using automated tools

### 2.1 Manual Testing

Manual testing is a type of software testing in which test engineers manually execute the test cases [7]. It is the most primitive of all testing types. In this testing tester initiates each test interacts with the system, reports, and evaluates the test results. To satisfy the test results test engineers should prepare and execute each test case properly.

**How to Do Manual Testing:**

*A. Understanding the Requirements:*

In order to do manual testing successfully, first, we need to understand the requirements of the application. By understanding the requirements, we will know what needs to be tested and what classifies a bug.

*B. Writing Test Cases:*

Once we understand the requirements, we can able to write test cases. Test cases guide the test engineer to test modules and different scenarios within the software. Writing good test cases is makes executing the test cases easy and ensures maximum test coverage.

*C. Executing the Test Cases:*

Once the test cases are written then we can execute the test cases manually and mark each test as "passed" or "failed", or "skipped". While doing manual testing, it is very important to write notes on what happens after the execution of a particular test case.

*D. Writing Good Bug Reports:*

After testing is done, the tester is responsible for writing a bug report. If a bug encounters the test engineer needs to give information to the development team about the bug. Writing good bug reports helps us to understand easily. A good bug report should have a strong title, expected and actual result, and any relevant attachments that will help the development team understand the issue (screenshots, files, etc...).

*E. Report on the Test Results:*

After running tests, we should know and write the results of the tests. How many tests are run? How many tests are failed? How many tests are skipped?

### 2.2 Continuous Testing using Automation

Automation testing is software which is used to test execution and the comparison of actual results with the expected results [8]. Automation software has different test data. Once the test case is automated, no human intervention is required. The aim of Automation is to reduce the number of test cases to be run manually and not to eliminate manual testing altogether. We can automate unit testing, integration testing, regression testing, black-box testing[9], etc.

**How to Achieve Continuous Testing using automation:**

*A. Test Tool Selection:*

Test tool selection depends on the technology the application is built on.

*B. Define the scope of Automation:*

The scope of automation defines the area of our Application under Test which will be automated. The following points will help to determine the scope:

(i) Scenarios which have a large amount of data
(ii) Common functionalities across the application
(iii) The complexity of test cases
(iv) Ability to use the same test cases for cross-browser testing

*C. Planning, Design, and Development:*

In this phase, we create an automation strategy and plan. It contains the following details

(i) Automation tools selected
(ii) Framework design and its features
(iii) In-Scope and Out-of-scope items of automation
(iv) Automation test bed preparation
(v) Schedule and Timeline of scripting and execution
(vi) Deliverables of Automation Testing

*D. Test Execution:*

In this phase, automation scripts are executed. These scripts need test data as input before set to run.

*E. Maintenance:*

As new functionalities added to the application automation scripts need to be added, reviewed, and maintained for each release. Maintenance is mandatory to improve the effectiveness of automation scripts.

### 2.3 Continuous testing using CI/CD

Continuous Testing phase involving with Build and Test phases [10] including Level 2 testing and Level 3 testing. Under the Level 2 testing, feature tests for current sprint, user stories tests will be prepared and the smoke tests will be completed. CT teams will work on generating test automation scripts from requirements. They will simulate test environment and access test data on demand. To ensure comprehensive cloud-based API testing, the open source testing tools will be integrated. Built-in automated application security testing will be activate over the cloud and promote artifacts when tests pass by orchestrating and automating the pipeline

**How to do Continuous Testing:**

*A. Create Test Environment:*

Create a test environment that means starting the testing process.

*B. Copy and anonymize production data to create test data:*

Production data is a copy of a database that has been masked. In this, we are using this production data to create test data[11].

*C. Use service virtualization to test API:*

API is a set of routines and tools for building software applications. API virtualization is a process of using tools that creates a virtual copy of our API, which mirrors all of the specifications of our production API and using this virtual copy in place of our production API for testing[12].

*D. Parallel Performance:*

Testing parallel testing is a software testing type, which checks multiple sub-modules of application concurrently to reduce the test time. In this, the tester runs two different versions of software concurrently with the same input.

### 2.4. Autonomous Testing

Testing of software using Machine Learning and Artificial Intelligence is called Autonomous testing[13] or smart testing. Practical implementation of smart testing is done in 6 levels same as self-driven cars. In each level, we use techniques to find bugs[14]

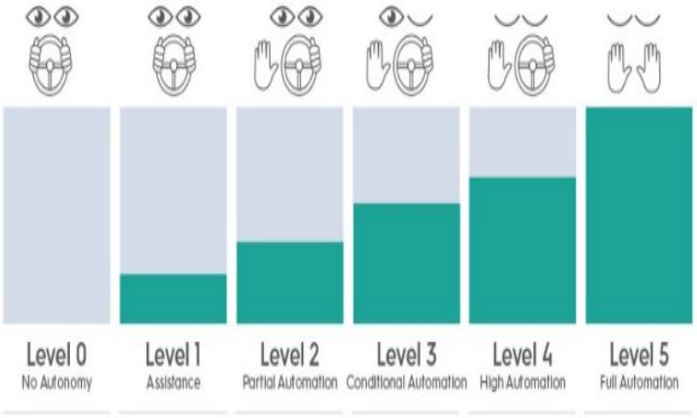


**Figure**.1. Levels of Autonomous testing

*Level 0: No autonomous*

Here, the human is writing the test and verifying each changing to the baseline.

*Level 1: Assistance*

The human is writing the test and verifying each change to the baseline.AI is assisting in writing test code like in a self-driven car[15] the human is still driving the car and the AI is only assisting while the human is driving. In this phase, AI can also check test pass or not but when the test fails it still needs to notify the human to check whether the failure is real one or happened because of the software after that human will have to confirm that whether it is a bug or software problem.

*Level 2: Partial Autonomous*

The human is writing the test and verifying each change to the baseline[16].AI is assisting the change verification by grouping changes like in self-driven cars human is still drives, but the AI takes care of acceleration and steering while assuming that the human can take over on level 1.Level 2 AI will be able to group these changes and tell the human that these are the same change, and they would like to please confirm or reject all these changes as a group.

*Level 3: Conditional Autonomous*

Human is writing the test.AI is verifying each change to the baseline like in a self-driven car where the car drives itself, but only under certain conditions, and always under the assumption that the human can take over when the AI notifies it that it is incapable of responding to a certain situation. In Level 2, if any failure or changes detected in the software still needs to be verified by a human. Level 2 AI can help to analyze the change, but cannot understand whether a page is correct or not just by seeing at the page. It needs a baseline to compare against, but Level 3 AI can do that and much more because it can apply machine learning techniques to the page. Now AI can look at pages and determine whether the page is OK or not without human intervention, just by understanding design and data rules and AI is looking hundreds of test results and seeing how things change and submit a test to humans to verify if such anomaly is detected.

*Level 4: High Autonomous*

Human is assisting in writing the test.AI is writing the test, with human guidance like in a self-driven car AI totally takes overall driving. Level 4 AI will able to understand pages by visually looking at the pages as a human does. Once AI understands the type of page (i.e. login page/profile page etc.) by using techniques reinforcement learning, it can write tests themselves

*Level 5: Fully Autonomous*

Human does nothing AI is writing the tests, without human guidance. In this level, AI will be able to converse with the product manager, understand the application, and fully tests itself.

## 3. Evolution of the testing Strategies

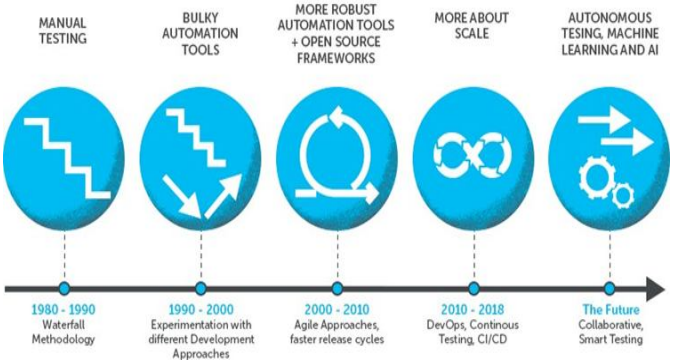


**Figure**.2. Evolution of the testing strategies

### 3.1 Manual testing test plan:

Here I am having manual testing test plan template. Generally the test plan has to prepare by test manager[17].

**Project Name:**

**Test Case Template**

**Test Case ID:** Fun_10
**Test Priority (Low/Medium/High):** Med
**Module Name:** Google login screen
**Test Title:** Verify login with valid username and password
**Description:** Test the Google login page

**Test Designed by:** <Name>
**Test Designed date:** <Date>
**Test Executed by:** <Name>
**Test Execution date:** <Date>

**Pre-conditions:** User has valid username and password
**Dependencies:**

| Step | Test Steps | Test Data | Expected Result | Actual Result | Status (Pass/Fail) | Notes |
|---|---|---|---|---|---|---|
| 1 | Navigate to login page | User= example@gmail.com | User should be able to login | User is navigated to | Pass | |
| 2 | Provide valid username | Password: 1234 | | dashboard with successful | | |
| 3 | Provide valid password | | | login | | |
| 4 | Click on Login button | | | | | |

**Post-conditions:**
User is validated with database and successfully login to account. The account session details are logged in database.

**Figure**.3. Sample Test case Template

*Example Scenario:* Based on the above template, below is an example that showcases the concepts in a more understandable way. Based on the above template, below is an example that showcases the concepts in a more understandable way

**Table 1.** Real-time Test case – Login – Positive test case.

| **Test Scenario ID** | Login-1 | **Test Case ID** | Login-1A |
|---|---|---|---|
| **Test Case Description** | Login – Positive test case | **Test Priority** | High |
| **Pre-Requisite** | A valid user account | **Post-Requisite** | NA |

Test Execution Steps:

| **S.No** | **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | **Test Result** | **Test Comments** |
|---|---|---|---|---|---|---|---|
| 1 | Launch application | https://www.facebook.com/ | Facebook home | Facebook home | IE -11 | Pass | [Satya 10/09/2020 11:44 AM]: Launch successful |
| 2 | Enter correct Email & Password and hit login button | Email id : test@xyz.com Password: ****** | Login success | Login success | IE -11 | Pass | [ Satya 10/09/2020 11:45 AM]: Login successful |
| | | | | | | | |

Suppose you are testing the login functionality of any web application, say Facebook. Below are the test cases for the same:

**Table 2.** Real-time Sample Test case – Login – Negative test case

| **Test Scenario ID** | Login-1 | **Test Case ID** | Login-1B |
|---|---|---|---|
| **Test Case Description** | Login – Negative test case | **Test Priority** | High |
| **Pre-Requisite** | NA | **Post-Requisite** | NA |

Test Execution Steps:

| **S.No** | **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Browser** | **Test Result** | **Test Comments** |
|---|---|---|---|---|---|---|---|
| 1 | Launch application | https://www.facebook.com/ | Facebook home | Facebook home | IE -11 | Pass | Satya 10/09/2020 11:44 AM]: Launch successful |
| 2 | Enter invalid Email & any Password and hit login button | Email id : invalid@xyz.com Password: ****** | The email address or phone number that you've entered doesn't match any account. Sign up for an account. | The email address or phone number that you've entered doesn't match any account. Sign up for an account. | IE -11 | Pass | [Satya 10/09/2020 11:45 AM]: Invalid login attempt stopped |
| 3 | Enter valid Email & incorrect Password and hit login button | Email id : valid@xyz.com Password: ****** | The password that you've entered is incorrect. Forgotten password? | The password that you've entered is incorrect. Forgotten password? | IE -11 | Pass | [Satya 10/09/2020 11:46 AM]: Invalid login attempt stopped |

### 3.2Real-timeAutomation testing test plan

This section is related to the automating the manual Smoke and Regression test cases[18] for a customer. This plan contains project specific Automation Test Plan information which includes the following:

*Scope of Automation:* The manual Smoke and Regression tests cases of Enrollment, Disease Management (DM) and Device Management (DVM) will be automated.

*Timeline:* Deliverables and schedule

*Test Strategy:* Testing techniques, entry and exit criteria, tools, test environment, test data

*QA team:* Organization, roles and responsibilities, training requirements, inter-group coordination.

***Automation Coverage:***

**Table 3.** Real time Sample Test case - Automation Test Coverage module.

| **Modules** | **Yes/No** |
|---|---|
| Enrollment | Yes |
| Disease Management | Yes |
| Device Management | Yes |
| Smoke test cases | yes |

Traceability Matrix will be maintained by mapping Requirements onto test scripts[19]: This will be used to track Test Case to Test Script (one to one mapping).

**Functional Requirements – Criticality Guidelines:** The criticality of each Test Script is defined by the Test Cases steps.

**Table 4.** Real-time Sample Test case – Functional Requirements.

| Sr. No. | **Functional Requirement** | **Criticality** |
|---|---|---|
| 1 | Test Case steps less than 4 steps | Low |
| 2 | Test Case steps less than 8 and greater than 4 steps | Medium |
| 3 | Test Case steps greater than 8 steps | High |
| 4 | Data Base verification | High |

The Regression and Smoke test cases are in scope and rests of the test cases which do not fall under regression are considered to be out of scope

**Assumptions:** The plan is based on the following assumptions which affect scope of testing:

- Application should be stable.
- The Manual Regression and Smoke test cases Test Cases of Enrollment, Disease Management and Device management will be automated.
- The requirements are complete, accurate and are testable.
- Any change in the scope of the project will be done after impact analysis and will be accompanied by changes to the project schedule/plan.
- QTP (QuickTest Professional) HP product licenses are available for 3 resources.
- VPN Connections are available for 3 resources.
- Environment setup and build deployment will be taken care by respective team not by automation team.

***Test Deliverables and schedule:*** The following table lists all the major QA deliverables with their due dates. A deliverable may contain more than one document. Details of schedule need to be defined

**Table 5.** Real-time Sample Test case – Test Deliverables and schedule.

| Deliverable | Description | Start Date | End Date |
|---|---|---|---|
| Automation Test Plan | This describes the overall test approach and strategy. | 09-Sep-2020 | 09-Sep-2020 |
| Automatable Test Cases | Automatable Test Cases | | |
| Test Scripts | Test Scripts | | |
| Test Report | Test Report | | |

- **Testing approach – Smoke Testing**

  *Objective:* Smoke testing is done by using automated smoke test scripts to make sure the build is stable and further testing cab be performed.

  *Entry Criteria:*
  - QA environment is setup with necessary hardware and software
  - Test scripts are designed and reviewed

  *Exit Criteria:* The entire smoke test Scripts are passed

  *Test Suspension and Resumption Criteria:* Testing will be suspended if:
  - Showstoppers or critical issues, which prevent the testing of major functionality, are encountered.
  - The test scripts fail.

  *Special Considerations:* Special considerations will be noted here.

- **Testing approach – UI/ Usability Testing**

  During the automation testing, Usability testing will not be done by developers. .

- **Testing approach – Functional Testing**

  During the Functional testing, the functional requirements and specifications need to be tested by the teams.

- **Testing approach – Regression Testing**

  *Objective:* Regression testing is performed to check that the previous release is not affected by the changes made in the current release. Regression testing is done by executing

the test Scripts from each release, which is a subset of the functional test cases for each release.

*Entry Criteria*

- Application is successfully installed
- Test Scripts are ready to be executed.
- Smoke test is successful
- Defects in previous release have been addressed according to the functional requirements.
- Developer release notes are provide with scope of testing, known issues, list of defects fixed, information about other parts which might have been affected by the defect fixes.

*Exit Criteria*

- No critical and high severity defects are open
- All known issues are documented (with workaround, if necessary)

*Test Suspension and Resumption Criteria:* Testing will be suspended if:

- Showstoppers or critical issues, which prevent the testing of major functionality, are encountered.
- Functionality is unstable, i.e., too many non-reproducible defects are encountered.
- Testing will be resumed when the showstoppers are fixed.

*Special Considerations:*

Special considerations will be noted here.

**Test Automation Prerequisites:** Automation team would be using Integrated Test Automation[20] Framework (ITAF) which has the below Key features

**Key features of ITAF**

- Reduces redundancy and increases reuse
- Reduces time to develop automation suite
- Robust and highly scalable
- Requires minimum maintenance
- Highly portable
- Supports Keyword and Data driven approach
- Grouped by function categories
- Extensible across domains
- Can be used by non-technical users to execute tests and monitor results without tool proficiency
- Facilitates customized reporting
- Functions, suites, applications
- Delivered through web, e-mail and other formats

The main execution of the application test starts from driver script. Driver script invokes QTP and loads Library files (Business, Generic and test scripts) which are mapped to it. Test data is invoked as and when the data is required to the script and test results are stored in test results folder

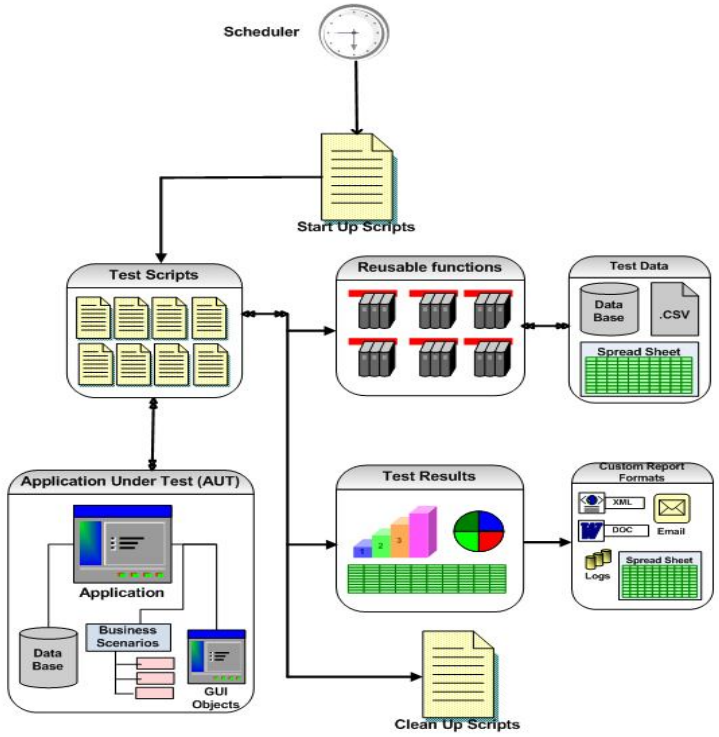


**Figure. 4.** Real time Integrated Test Automation Framework (ITAF)

ITAF consists of the below folder structure which would be used as it is.

- *Script Folder:* QTP script that drives the application testing
- *Environment File in Text format:* The component's path is saved in this file which is in turn called from driver script
- *Library in Vbs format:* These are external "VBS" files that are added to the resources tab of the QTP test settings. This file will contain public functions of specific Business scenarios and Generic scenarios that can be reused across the test scripts. These library files would be stored in the folder "Library"
- *Object Repository:* When you create a test or component, all the information about the objects in your test or component is stored in an object repository. You can view and modify this information in the Object Repository dialog box
- *Data table:* To retrieve test data required by the test scripts most popularly used Data tables are Excel data tables
- *Recovery and Exceptional Handling:* Recovery Scenarios are used to recover from failures or any unexpected events. On Error Resume is used in the functions for exceptional handling.
- **Criteria for Inclusion of Test Cases in Automation[21]**
  - Application is successfully installed.
  - Smoke testing is successful.
  - Test scripts are ready to be executed.
- *Entry Criteria:*

• No critical and high severity defects are open in the functionality related to automated scripts.

• All applicable parameter data are in place.

- *Test Suspension and Resumption Criteria:* Testing will be suspended if:

• Showstoppers or critical issues, which prevent the testing of major functionality, are encountered.
• The test scripts fail.

*Special Considerations:* If any special considerations need to be added here

### 3.3 Continuous testing test plan

The continuous testing plan[22] contains project-specific Quality Assurance Test Plan (QATP) information for prepare and conduct system & regression testing which includes the following:

*Scope of testing:* Sanity& regression test cases will be automated along with manual testing for all the requirements

*Source of Information and Timeline:* Define sources of information used to prepare the plan deliverables and schedule

*Test Strategy:* Testing techniques, entry and exit criteria, tools, test environment, test data

*QA team:* Organization, roles and responsibilities, training requirements, inter-group coordination

*Text execution:* Test metrics, Test Report, and Defect management

***Objectives:*** Quality objectives of the Testing are to ensure complete validation of the business and application requirements

- Verify application requirements are complete and accurate
- Perform detailed test planning
- Identify testing standards and procedures that will be used on the project
- Prepare and document test scenarios and test cases
- Regression testing to validate that unchanged functionality has not been affected by changes
- Manage defect tracking process
- Provide testing execution and summary reports

***Reference Documents:***

- HLD: define a path for High-level design
- LLD: define a path for Low-level design
- Wireframe: define a path for wireframes
- Business Requirement: define a path for business requirements

**Real time Automation Framework:** An Automation Framework is an integrated platform which hosts various hardware and software tools, resources and services. It enables efficient design and development of automated test scripts and analysis of issues for the system under test[23].

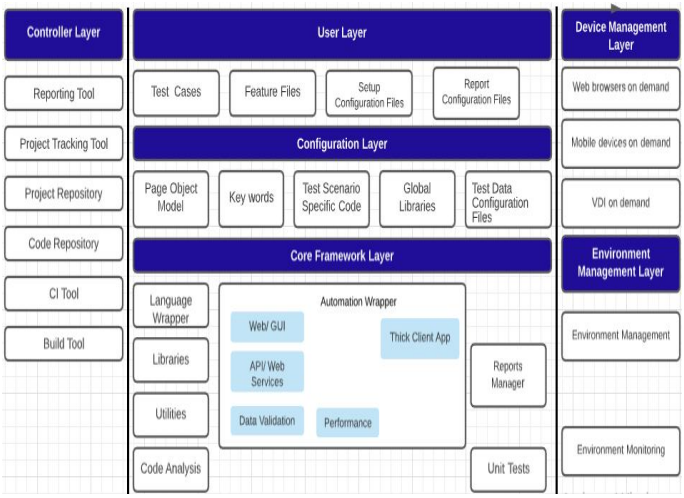


**Figure. 5.** Real time Automation Framework – Conceptual View

**Scope of Continuous testing:** The high-level modules that will be tested are

- Operations
- Maintenance
- Quality Analysis
- Inquiry
- Configuration
- Utilities
- Reports
- Administration

These modules are derived from the basic workflow of the system. Apart from these modules, Inbound and Outbound interface files will be tested. Capacity planning, performance tuning, network configurations, code changes, and Database procedure changes are considered as out of scope.

**Continuous testing – Test Approach:** After the Project Increment (PI) planning, the Product Owner (PO) will develop feature files[24] for each story in the respective sprints and will be handed over to QA team. QA teams will prepare the manual test cases and share them with the respective PO. Once the manual scripts validations are done, the QA team starts preparing the skeleton for the automation test scripts [25] based on the availability of wireframes. Meanwhile, the development teams will develop and diagnostics the code in the process of unit testing before deploying to the test environment. For every new build sanity test job will be automatically triggered into integrated tools like Jenkins and based on sanity results QA teams will decide whether to accept or reject the build. Once the build is accepted, QA teams will test the deployed stories and report the bugs in version and will be tracked till closure[26]. Version tool will be used for logging and tracking the defects rose as part of functional testing and tracked to closure. Regression test cases will be identified by QA teams and end-to-end (E2E) automation will be done. Once the testing is done in the test environment the same E2E testing will be carried out in stage.

**Different types of testings' covered in Continuous testing:**

*Sanity Testing:* Sanity testing will be done by using automated test scripts[27] to check the health of the application to make

sure the build is stable, sanity testing will be performed after each build same will be integrated to CI/CD pipeline.

*System Testing/InSprint testing:* system testing will be done during InSprint[28] once user stories are deployed in the test environment and sanity is pass to make sure all the user stories are developed as per requirements if any deviation found defects will be logged in Version and tracked till closure.

*Regression Testing:* Regression testing will be conducted before every release to make sure current sprint development didn't affect existing functionality. Regression test cases will be identified during manual test design and the same will be automated before adding those to the existing regression suite.

*Performance Testing:* Ensuring that no performance degradation is witnessed com-pared to the SLA's or relative to the previous iteration of performance testing [29]carried out. Measure, record, and summarize all performance measurements (both successful and unsuccessful) that occur during testing. Collation of all results will be analyzed and used as part of the detailed performance test exit report.

**Table 6.** Real-time Performance Testing Check List

| | |
|---|---|
| **Smoke Testing** | Yes |
| **UI Testing** | Yes |
| **Functional Testing** | Yes |
| **Regression Testing** | Yes |
| **Test Automation** | Yes |
| **Performance Testing** | Yes |

- **Continuous Testing - Test Deliverables**
  - Test Plan
  - Manual Test Cases
  - Automation Test Scripts
  - Defect/Enhancement Logs
  - Test Reports
  - Final Performance closure document

**Table 7.** Real-time Continuous Test Environment – Tools to be used

| No | Purpose | Tool |
|---|---|---|
| 1 | Source code Management | GitHub |
| 2 | Test Script Development and Execution | Selenium, Java, BDD, Eclipse, Jenkins, Maven, Extent reports, Jenkins |
| 3 | Performance Testing | Load Runner |

**Real Time Test Metrics and Defect Management:**

- *Test Metrics:* The following test metrics will be used to monitor test progress and product quality[30]. The metrics must be correlated to gain more understanding. The metrics will form part of project QA sheet.

**Table 8.** Real-time Test Metrics table

| Metric Name | Description |
|---|---|
| Time to find a defect | Time spent on testing/total number of defects found during that time |
| Severity Index | Weighted average of defects logged |
| Defect Age | Time to fix the defect |
| Defects to Remarks Ratio | Number of defects against the number of remarks |

- *Test Reporting:*
  - Manual test cases will be shared in Excel
  - Automation test cases will be shared as Behavior Driven Development (BDD) feature files
  - Automation test results will be shared
  - Weekly and Daily test results will be shared to respective stake holders
  - Final performance closure document will be shared
- **Defect Management:**
  - Versions will be used as defect management tool in current project[31]
  - QA team will report all the deviations from the use cases and the validation and verification issues into the defect management system
  - Defect logged by QA team and then assigned to development team for further investigation, whole team is responsible to track defect till closure.
- *Guidelines for entering defects:*
  - Defect title should be clear and convey the issue
  - Defect description should explain all the details about the defects
  - Steps to reproduce should be included in the description along with actual and expected behavior.
  - Proper screen shots should be attached while logging the defects
  - QA team need to select priority as per the guidelines
  - QA team need to select severity as per the guidelines
  - QA team should assign to developer if it is inSprint and regression defects should assigned to common bucket and scrum masters will assign those to developers based on availability
  - QA team and Developers should follow defined bug life cycle and update the defect information in versions
  - QA team need to attach proper screenshots after retesting the defect
- **Priority and Severity guidelines:** Defects are classified on two scales. Those are severity scale and priority

scale[32].

- *Severity Scale:* The severity scale indicates the impact of a failure caused by a fault. QA engineer responsible for entering the defect and he will decide the severity. It can be classified into the following:

**Table 9.** Real-time Priority and Severity guidelines

| Severity | Guidelines |
|---|---|
| Critical | System Crash<br>Defects that have the potential to destabilize the system<br>Missing major functionality<br>Any defect that prevents execution of test cases |
| High | Erroneous functionality<br>Loss or corrupted data transformation<br>Unexpected functional behavior<br>Wrong/missing features in functionality<br>Major non-conformance to GUI standards<br>Workaround exists but it is complicated |
| Medium | Errors in the display of graphics and content<br>Unexpected functional behavior which are not showstoppers<br>Minor non-conformance to GUI standards<br>Defect in code, but simple workaround exists |
| Low | Cosmetic errors<br>Defect causes minimal or unnoticeable problems |

- *Priority Scale:* The priority scale indicates the urgency to fix a fault. Priority is decided by the project manager or an individual or a group appointed by him/her. Guidelines for priority are listed below:

**Table 10.** Real-time Priority scale Guidelines

| Priority | Guidelines |
|---|---|
| High | Resolve immediately |
| Medium | Resolve Pre-Release |
| Low | Desired but not urgent |

**Advantages of Autonomous Testing**

- Accelerating manual testing and the overall process.
- Automation of the testing process.
- Eliminating more bugs
- Reduced ignore bugs probability.
- Forecasting client requirements.

## 5. CONCLUSION

Here I have identified the research scope in autonomous testing for real time IOT based application[33]. While using machine learning and artificial intelligence in autonomous testing, the huge amount of data and systems involved is greater than the data needed for automation. It aggregates the data that will be dynamic which practiced from the machine learning algorithms.

## REFERENCES


1. M. Pouse, "Artificial Intelligence," 09 09 2020. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence. [Accessed 19 09 2020].
2. K Chaitanya, et al. “A Formal and Enriched Framework for Testing Distributed Embedded Systems.” International Journal of Emerging Trends in Engineering Research, no. 9, The World Academy of Research in Science and Engineering, Sept. 2020, pp. 6389–96. Crossref, doi:10.30534/ijeter/2020/238892020.
3. F. L. Bauer, "Software and Software Engineering," SIAM, vol. 15, no. 2, pp. 469-480, 2006.
4. S. N, "Key Metrics Identification of Distinct Process Models to develop an IOT based systems," IJRTE, vol. 8, no. 4, pp. 9661-9666, 2019.
5. P. GACZKOWSKI, "Bridging Gaps: The Importance of DevOps Communication," Developers, [Online]. Available: https://www.toptal.com/devops/bridging-gaps-devops-communication. [Accessed 2020 8 10].
6. D. Solutions, "Software Testing," DroIT Solutions, Pune, 2018.
7. A. Choudary, "Manual Testing Complete Guide: Everything You Need To Know," Edureka, 28 Nov 2019. [Online]. Available: https://www.edureka.co/blog/what-is-manual-testing/. [Accessed 10 8 2020].
8. Panda B.S., Suman R.S., Hemanth A., Kumar D.H. (2019), ‘Model-based automation in testing of web applications’, International Journal of Innovative Technology and Exploring Engineering, 8(7), PP.3056-3062.
9. Jammalamadaka K., Parveen N. (2019), ‘Holistic research of software testing and challenges’, International Journal of Innovative Technology and Exploring Engineering, 8(0), PP.1506-1521
10. A. Eran Kinsbruner, " Manual Testing vs. Automated Testing.," Perfecto., 13 August 2019. [Online]. Available: https://www.perfecto.io/blog/automated-testing-vs-manual-testing-vs-continuous-testing. [Accessed 01 Sep 2020]..
11. “Effective Automated Testing Strategies for Release Pipelines | CloudBees Docs.” *CloudBees Docs*, https://docs.cloudbees.com/docs/cloudbees-cd/latest/pipelines/leverag-test-data-mgmt. Accessed 27 Sept. 2020.

12. "Continuous Test Data - DATPROF." *DATPROF*, 11 May 2020, https://www.datprof.com/solutions/continuous-test-data/ .
13. Philip Koopman, and Michael Wagner. "Challenges in Autonomous Vehicle Testing and Validation ." *2016 SAE World Congress*, 2016 SAE World Congress, Jan. 2016, doi:2016-01-0128 / 16AE-0265.
14. J. M. Zhang, M. Harman, L. Ma and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons," in *IEEE Transactions on Software Engineering*, doi: 10.1109/TSE.2019.2962027.
15. Gil Tayar. "Not Only Cars: The Six Levels of Autonomous Testing - Automated Visual Testing | Applitools." Automated Visual Testing | Applitools, 24 Oct. 2017, https://applitools.com/blog/not-only-cars-the-six-levels-of-autonomous/.
16. "The 6 Levels of Vehicle Autonomy Explained | Synopsys Automotive." Synopsys | EDA Tools, Semiconductor IP and Application Security Solutions, Synopsys, Inc, https://www.synopsys.com/automotive/autonomous-driving-levels.html. Accessed 28 Sept. 2020.
17. Bhat, Gururaj. "Sample Software Test Plan Template with Format and Contents." Software Testing Help - Free Software Testing & Development Courses, https://www.softwaretestinghelp.com/test-plan-template/. Accessed 28 Sept. 2020.
18. smita. "How to Select Correct Test Cases for Automation Testing (and Ultimately Achieve a Positive Automation ROI)." Software Testing Help - Free Software Testing & Development Courses, https://www.softwaretestinghelp.com/manual-to-automation-testing-process-challenges/. Accessed 28 Sept. 2020.
19. Shilpa. "4 Simple Steps to Create Requirement Traceability Matrix (RTM) - Free Sample to Download | Opencodez." Opencodez, 8 Feb. 2017, https://www.opencodez.com/software-testing/create-requirement-traceability-matrix-rtm-free-sample-download.htm.
20. Kirsten Aebersold. "Test Automation Frameworks." Smartbear.Com, SmartBear Test complete, https://smartbear.com/learn/automated-testing/test-automation-frameworks/. Accessed 28 Sept. 2020.
21. Flemström, D., Potena, P., Sundmark, D. et al. Similarity-based prioritization of test case automation. Software Qual J 26, 1421–1449 (2018). https://doi.org/10.1007/s11219-017-9401-7
22. Tom Alexander. "DevOps Testing Strategy: Benefits, Best Practices & Tools." Test Management Tools | Jira Test Case Management Software | Zephyr, SmartBear Software, 30 Apr. 2018, https://www.getzephyr.com/insights/developing-devops-testing-strategy-benefits-best-practices-tools.
23. Raju, Devendra. "Selenium Automation Framework Example | Selenium Easy." Selenium Easy, Selenium Easy, 11 May 2014, https://www.seleniumeasy.com/selenium-tutorials/selenium-automation-framework-example.
24. "DevOps - Scaled Agile Framework." Scaled Agile Framework, Scaled Agile, 20 Jan. 2019, https://www.scaledagileframework.com/devops/.
25. Sasi Bhanu J., Baswaraj D., Bigul S.D., Sastry J.K.R. (2019), 'Generating test cases for testing embedded systems using combinatorial techniques and neural networks based learning model', International Journal of Emerging Trends in Engineering Research, 7(11), PP.417-429
26. Alexander Granada Murdoch, and Ahmad Nurul Fajar. "Designing Security Testing Systems Integration Using Service Oriented Architecture." International Journal of Emerging Trends in Engineering Research, no. 9, The World Academy of Research in Science and Engineering, Sept. 2020, pp. 6389–96. Crossref, doi:10.30534/ijeter/2020/238892020.
27. Vijay Kumar. "Smoke Testing vs Sanity Testing vs Regression Testing Explained." Testsigma Blog, 7 Feb. 2020, https://testsigma.com/blog/smoke-testing-vs-sanity-testing-vs-regression-testing-explained/.
28. Scot Noftz. "5 Steps to Succeed at In-Sprint Test Automation - SPR." SPR, 29 May 2019, https://spr.com/5-steps-to-succeed-at-in-sprint-test-automation/.
29. "Performance Testing - The Complete Guide - Neotys." Neotys, http://www.facebook.com/Neotys, https://www.neotys.com/insights/performance-testing. Accessed 29 Sept. 2020.
30. Subramanian, Santosh. "Software Testing Metrics Driving Testing Services Performance." Whitepaper_Software-Testing-Metrics, Mphasis, https://www.mphasis.com/content/dam/mphasis-com/global/en/downloads/WhitePapers/Whitepaper_Software-Testing-Metrics.pdf. Accessed 29 Sept. 2020.
31. Swati Seela, and Ryan Yackel. "64 Essential Testing Metrics for Measuring Quality Assurance Success | Tricentis." Tricentis, Tricentis, 27 Jan. 2016, https://www.tricentis.com/blog/64-essential-testing-metrics-for-measuring-quality-assurance-success/.
32. Kirandeep Kaur. "Bug Severity vs Priority In Testing With Examples." LambdaTest, Lambda Test, 8 Apr. 2019, https://www.lambdatest.com/blog/bug-severity-vs-priority-in-testing-with-examples/.
33. Pandit D.P., Pattanaik S.R. (2019), 'Software engineering oriented approach for iot applications: Need of the day', International Journal of Recent Technology and Engineering, 7(6), PP.886-895