LEANBIOLOGIX

IT OPERATIONAL EXCELLENCE FOR LIFE SCIENCE MANUFACTURERS

Automated Computer Software Assurance (ACSA) in 2025

*Top players are **fully automating** the computer system validation process.*

www.leanbiologix.com

(508) 541-6383

inquiries@leanbiologix.com

Overview

This document is meant to be used a guide to help life sciences manufacturers to **understand** and **implement** Automated Computer Software Assurance (ACSA).

This guide it NOT a copy-paste reiteration of industry manuals, it has been written from the perspective of IT leaders and engineers who have experienced the various transformations of the industry over the last 10 years. It also focuses on the new paradigm of automated testing solutions including Selenium integrations and ALM solutions.

To give the reader context, sections are included summarizing the principles of computer system validation (CSV) and GAMP guidance. This book aims to be the go-to resource to senior leadership, IT Quality & Compliance, system owners, compliance and technical analysts.

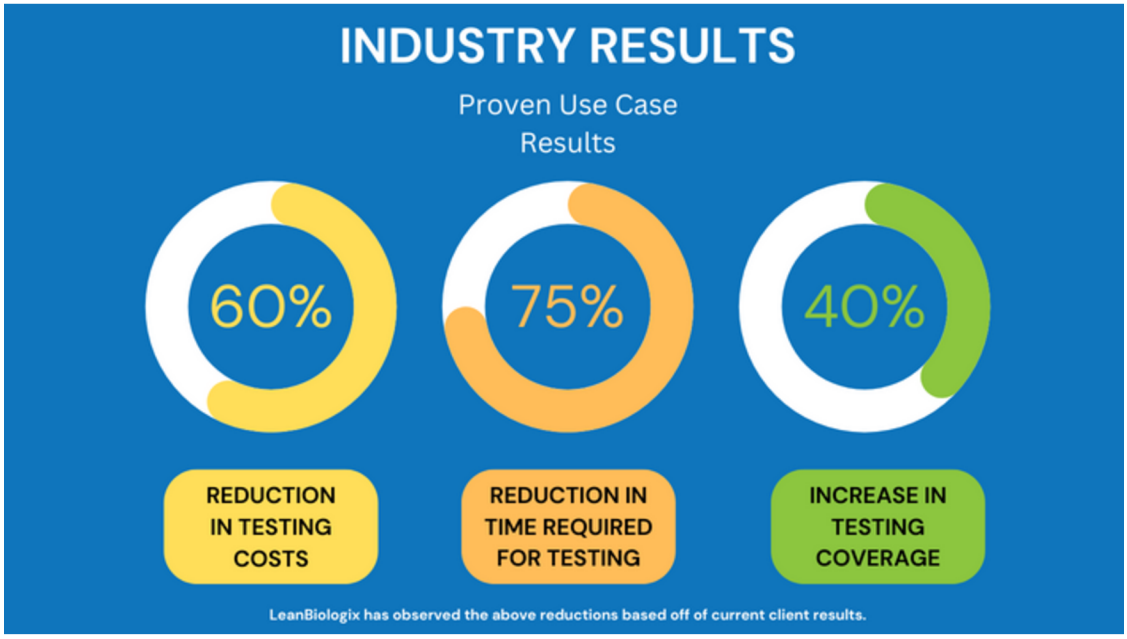
CASE STUDY (click me) | Value of Automated Testing | GAMP5 Connection | Implementation Examples

Section 1: Cost and Value - Implementing Testing Automation
- Explores time & cost savings, provides data insights through case study review

Section 2: Automated Computer Software Assurance (ACSA)
- Details modern ACSA implementation with a step-by-step integration guide

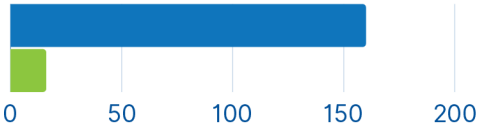
Section 3: EXAMPLE - Regulatory Compliance and Case Study for a CMMS System
- Examines 21 CFR Part 11 requirements and maps them to automated test cases

Case Study 1 Details



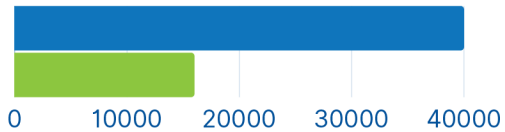
Results - Large (650M+) Medical Device Manufacturer

Effort Hours (yearly)



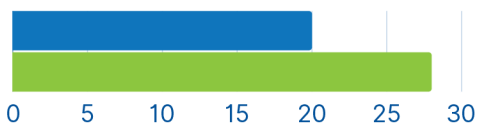
Annually there are 8-10 software patches which require risk-based regression testing. Each change takes about 40 hours including risk assessment, testing, document review and approvals.

Annual Cost (USD)



Initially the annual costs were $40000, after automated implementation annual cost was reduced to $16000.

Test Requirement Coverage



Test coverage went from 20 requirements run every month to 28 requirement tests run daily.

To illustrate the impact of automated testing, let's examine the following case study:

Large (650M+) Medical Device Manufacturer

- **Background**: Manufacturer faced challenges in validating routine monthly equipment management software updates, which was subject to regulatory requirements.
- **Challenge**: Manual regression testing was labor-intensive and lacked full coverage due to personnel availability.
- **Solution**: They adopted an automated testing approach with Selenium
- **Outcome**: Regression testing time was cut by 90%, and test coverage increased by 40%. The automation framework ensured that all regulatory requirements were met consistently.

Case Study Details - continued

Initial Process

A medical device company using multi-tenant SaaS Equipment management system was tasked with recurring testing on system requirements from frequent monthly upgrades.

The system has many customized business and functional requirements which require regression testing, as the software provider cannot confirm if the changes will impact a specific customers build.

The company therefore had to review each patch, create a test plan and associated change control documentation, perform testing and then get everything approved on a monthly cadence.

Before Automated Implementation

Process Improvements

Through implementation of validated automated system testing, the process was simplified from 4 steps to 2 steps. This lead to savings in cost, time, and an increase in compliance.

After Automated Implementation

Outcome - Automated Testing & Test Report Generation for all requirements

LEANBIOLOGIX

System	Test Script(s) Passed	# Skipped	# Failed	Total Time
Dynamics Testing	1	0	0	87.2 seconds

Result	Test Case	Test Script	Start	End
Dynamics Testing — passed	Automated_Test_1	Automated_Test_Script_1	11:00:25	11:01:52

Dynamics Testing

Test Case	Browser	URL	Test Executor User ID
Automated_Test_1	edge	https://businesscentral.dynamics.com/?noSignUpCheck=1	nrodgers@leanbiologix.com

Step #	Expected	Actual	Timestamp
1	Go to website: https://businesscentral.dynamics.com/?noSignUpCheck=1	Website access success.	29-Mar-24 11:00:30
2	Click 'Items' tab.	Items tab clicked successfully.	29-Mar-24 11:00:39
3	Click new record.	Clicked new record button.	29-Mar-24 11:00:45
4	Fill out required fields on new record form for Items.	Filled out all required fields on new record form for Items.	29-Mar-24 11:01:00
5	Take a screenshot.	Screenshot captured:	29-Mar-24 11:01:01

Chapter 1
The Case for Automated Testing

Introduction

In this chapter we explore the benefits of automated testing, focusing on its impact on time and cost savings over manual testing.

Automated testing offers a transformative solution by **significantly reducing** the **time** and **effort** required to validate a software system while enhancing the accuracy and reliability of test results.

By leveraging Automated Computer Software Assurance (ACSA) practices, organizations can get the following benefits:

- streamline their testing processes
- ensure regulatory compliance
- achieve a higher standard of quality
- reduce testing resource man hours

Ensuring compliance and maintaining system integrity are critical tasks for life science companies.

Traditional manual (electronic and paper-based) testing methods can also be thorough, but are often resource-intensive, time-consuming, and introduce human error. Testing Automation is the solution.

Value of Automated Testing

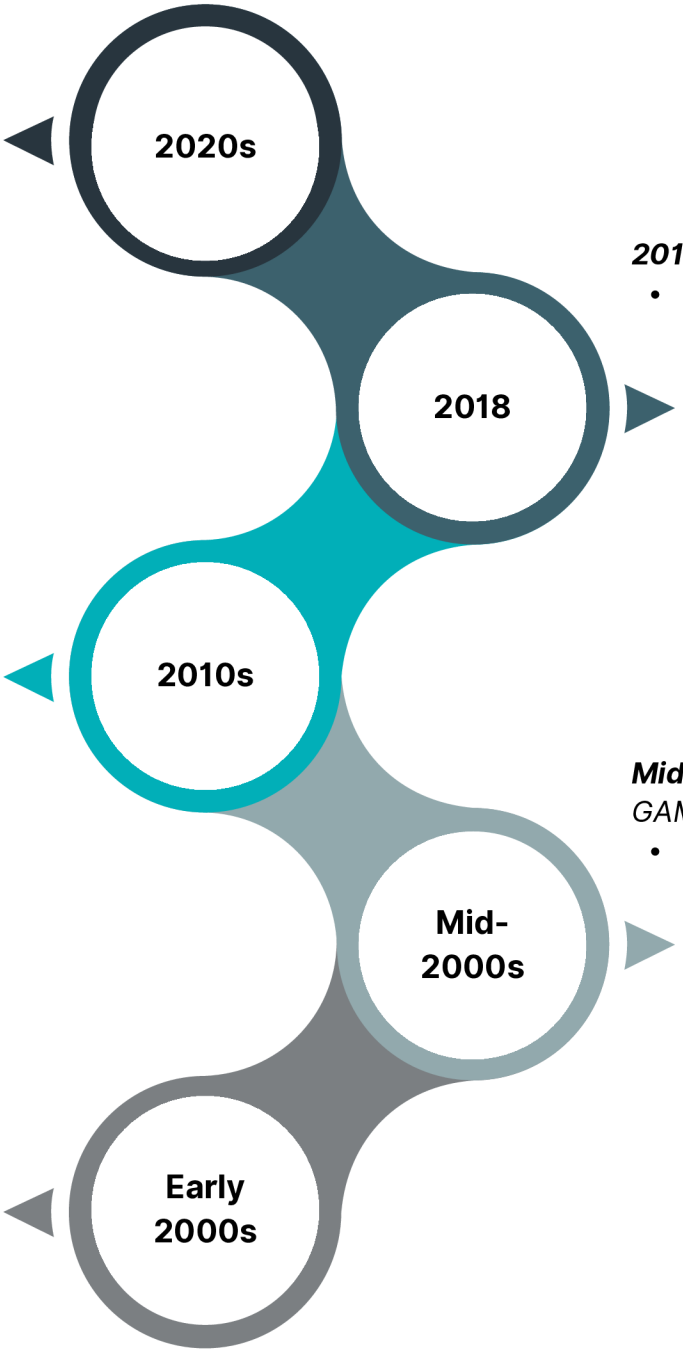
Automated testing provides numerous advantages over manual testing. The primary benefits are:

1. **Reduced Testing Time:** Automated tests can be executed much faster than manual tests, allowing for more frequent and comprehensive testing cycles. This reduction in testing time directly translates into faster project completion and quicker time-to-market for new systems and updates.
2. **Cost Savings:** By automating repetitive and time-consuming tasks, organizations can significantly reduce the labor costs associated with manual testing. Automated testing tools, once set up, can run multiple tests simultaneously and require minimal human intervention.
3. **Increased Test Coverage:** Automated testing enables broader test coverage by allowing for the execution of a larger number of tests across various scenarios and configurations. This comprehensive coverage helps identify defects and vulnerabilities that might be missed in manual testing.
4. **Improved Accuracy and Consistency:** Automated tests eliminate the risk of human error, ensuring that tests are performed consistently and accurately each time they are executed. This reliability is crucial for maintaining compliance with stringent regulatory requirements.
5. **Enhanced Reporting and Documentation:** Automated testing tools provide detailed reports and logs of test execution, which are essential for demonstrating compliance during audits. These tools also facilitate the generation of comprehensive documentation, reducing the burden on project teams.

2020s - Automated Testing, AI, and Machine Learning

2020s *- Integration of AI and Machine Learning*

- *Testing Techniques:*
  - *Specialized testing for AI/ML models.*
  - *ACSA - Automated Testing Frameworks: Advanced frameworks for dynamic continuous testing.*
  - *Algorithm Validation: Rigorous testing of AI/ML algorithms.*



2018 *- FDA's CSA Initiative*

- *Testing Techniques:*
  - *Risk-Based and Critical Thinking: Focus on critical system aspects.*
  - *Automated Testing and Continuous Validation: Streamlined processes with automation.*

2010s *- Shift Towards Cloud Computing*

- *Testing Techniques:*
  - *Automated Testing: Increased use of automated tools for complex cloud systems.*
  - *CI/CD Integration: Testing integrated into CI/CD pipelines for frequent and reliable testing.*

Mid-2000s *- Introduction of GAMP 5*

- *Testing Techniques:*
  - *Risk-Based Testing: Focus on high-risk areas as guided by GAMP 5.*
  - *Traceability Matrices: Ensure all regulatory requirements are tested.*

Early 2000s *- Adoption of 21 CFR Part 11*

- *Testing Techniques:*
  - *Manual Testing: Manual creation and execution of test cases.*
  - *Documentation-Driven Validation: Extensive documentation to demonstrate compliance.*

Modern techniques reflect the industry's need for efficient, risk-based, and ultimately automated testing methods to ensure compliance and product quality.

Chapter 2
GAMP5 (2022) Automated Testing Guidance

GAMP5 – Improved Guidance for Computer System Validation

The **GAMP5** second edition improved upon the original edition from 2008 and was released in July 2022.

The updated guide keeps the same structure but updates its content to highlight the growing area of cloud services. It also addresses the increasing use of **testing automation software** to maintain compliant systems.

Why revise ISPE GAMP5?
The timing of the new release is important as it coincides with the release of new FDA guidance "Computer Software Assurance for Production and Quality System Software".

This document provides additional guidance on validating computer systems. It explains how the industry should ensure quality in cloud services, new AI and infrastructure technologies, and automated system tests.

An important Highlight is the guidance acknowledgment of automated testing – "Using automated testing brings benefits to test coverage, repeatability, and speed." (Appendix D5 – Testing of Computerized Systems, section 25.1.1)

New technologies have been added as new Appendices. These technologies include Artificial Intelligence and Machine Learning (AI/ML), blockchain, cloud computing, and Open-Source Software (OSS).

Key GAMP5 Updates – Summary:
Modernization and Automation: The update highlights the integration of IT and cloud services, and the increased use of automation, aiming to enhance control, quality, and reduce risks.

Alignment with FDA Guidance: The release coincides with new FDA guidelines on computer software assurance, which helps the industry adapt to cloud services and automated testing.

Acknowledgment of Automated Testing: The updated guidelines recognize the benefits of automated testing such as improved test coverage, repeatability, and speed.

Inclusion of New Technologies: Technologies like AI/ML, blockchain, cloud computing, and open-source software are now considered, expanding the scope and applicability of the guidelines.

Modern Approach to GAMP5 Computer System Validation
Modern methods use data and documents generated by automated tools instead of traditional, detailed manual documentation for specifications and testing.

This approach leverages the capabilities of software tools to maintain records and information in a compliant way, and streamlines processes.

Automated Testing and the GAMP5 Test Management Process

Testing should focus on the system's intended use, with a clear link between test cases and requirements to ensure thorough coverage.

High-risk requirements with regulatory significance or direct product impact must be scripted for testing. Automate and reuse these high-risk scripted tests throughout the system's life cycle.

Agile development, common in both on-premise and SaaS software, allows for automated test cases to be created alongside code development or executed after a new version is made ready.

The scope and burden of this additional testing, depends on the automated tests' thoroughness.

Scripted testing (manual or automated) focuses on the software's intended use as defined by internal workflows, adjusting for risk levels of each requirement.

Even with detailed test coverage, there may still be a need for separate unscripted or scripted tests to assess end-to-end workflows and specific system use cases.

GAMP5 (2022) Automated Testing Guidance

Modernizing Software Testing using Critical Thinking – ISPE GAMP5 Updates

Critical thinking is a crucial component in the context of the updated ISPE GAMP®5 guidelines for several reasons explored below. It is necessary to note that Quality support roles, Compliance Specialists, and IT Admin should aim to eliminate unnecessary testing burden.

Summarized straight from the ISPE GAMP 5 guide:

Critical thinking fosters informed decision-making on applying and scaling quality and compliance for computerized systems. Success hinges on understanding business processes and analyzing their impact on patient safety, product quality, and data integrity.

Improved risk understanding leads to better risk control and robust scaling of controls and validation activities.

Practical Advice from Industry Experts:
Knowing how the business works and how the system functions is important for making the right decisions when testing software.

WHO is using the application and HOW the application being used is core consideration for risk assessment for functional requirements.

IT Quality and testing departments cannot truly assess product impact or risk without having a clear understanding of the process. Visual business process maps are important for project teams. They help align everyone at the start of a project, and make a significant impact.

Coordination with stakeholders and SMEs is required – but so is the individual compliance specialists/IT QA resources personal knowledge of the system and use cases.

SME resources help paint the full picture but they cannot make the final call on risk level as they often have roles which introduce a conflict of interest. For example “This is a low risk because if we assign it a high risk it’s going to increase my workload”.

The FDA guidance promotes the idea of **minimum viable coverage** (MVC) and emphasizes the importance of getting it right.

There are several areas impacted by misunderstanding MVC this including the following:

- Tester takes additional time to write and test in development
- Reviewers take additional time to review scope and steps
- Increased steps increase the chance of test script errors or defects during testing
- During execution, the testing timelines increase.
- Reviewers experience increased testing timelines post-execution.
- Periodic reviewers spend additional time reviewing the additional scope of testing
- Additional testing increased the burden of SDLC impact and increases review times
- During an audit, unnecessary steps hide important details that are not needed.
- Business users spend time doing unnecessary UAT testing
- Increased system downtime affects users and production schedules.
- Extra work places undue pressure on team members across the whole organization and increases project timelines.

The root cause of this can often be traced to a one-size-fits-all interpretation of compliant software testing strategy.

One solution is commitment of the project team to critical thinking and developing a non-superficial understanding of an application and the work being done.

Automated testing helps overcome the challenges of a one-size-fits-all approach to compliant software testing strategies by ensuring a thorough, efficient, and standardized testing process. This enables project teams to achieve a deeper understanding of the application and work being done, while also adhering to these MVC principles.

By committing to both critical thinking and automated testing, teams can enhance their compliance, reduce unnecessary burdens, and improve overall project outcomes.

GAMP5 (2022) Automated Testing Guidance

Leveraging Critical Thinking in Implementing GAMP5 Updates

The update aligns with new FDA guidance, which emphasizes modern approaches like **automated testing** and cloud services.

Critical thinking helps stakeholders understand these changes, evaluate their implications, and implement them effectively to meet regulatory requirements.

Adapting to Technological Advancements: With the integration of emerging technologies critical thinking allows us to assess how these technologies can be utilized within the accepted framework of computerized system validation.

This involves analyzing the risks, benefits, and potential impacts on quality and compliance.

Optimizing Test Management: The revised guidelines highlight the importance of linking test cases to requirements for thorough coverage, particularly for high-risk functionalities.

Critical thinking aids in discerning which aspects of the system require more rigorous testing and how automated tools can be leveraged to enhance test accuracy, repeatability, and efficiency.

Evolving Validation Strategies: Modern validation approaches now rely more on data and artifacts generated by automated tools rather than extensive manual documentation.

Critical thinking is necessary to evaluate the adequacy of these automated outputs in proving system validation and ensuring they meet all necessary specifications and regulatory standards.

Risk Management: The update to GAMP®5 emphasizes risk-based approaches to testing and validation, particularly for software directly affecting product quality or regulatory compliance.

Critical thinking is key in identifying potential risks, determining their severity, and deciding on the appropriate level of scripted testing and automation.

Enhancing Agility with Automated Testing

Automated testing can improve development processes by increasing efficiency, consistency, and speed. Here's how automated testing can be leveraged to support Agile methodologies:

Integration: Automated tests can be integrated into CI/CD pipelines, allowing for immediate feedback on code changes. This ensures that new code is quickly verified for integration issues, bugs, and compliance with requirements.

Deployment: Automated deployment processes can rapidly move tested code to production, reducing the time between development and release.

Rapid Feedback: Automated tests can be run frequently, even on each commit, providing rapid feedback to developers. This helps in identifying and fixing issues early, reducing the cost and time associated with bug fixing.

Regression Testing: Automated regression tests ensure that new changes do not break existing functionality, maintaining the stability of the product through continuous development cycles. Scalability: Automated tests can cover a large number of test cases, including those that are time-consuming or complex.

Consistency: Automated tests are consistent in their execution, reducing the risk of human error and ensuring reliable results across multiple test runs.

Automatic Documentation: Automated tests can generate logs and reports that document the testing process, providing a clear audit trail for compliance purposes.

Traceability: Automated tests should be linked to specific requirements, ensuring traceability from application function to test cases and test results.

GAMP5 (2022) Automated Testing Guidance

Best Practices for Implementing Automated Testing

Start with High-Value Tests:
Prioritize automating tests that provide the most value, such as critical functionality, high-risk areas, and repetitive tasks.

Maintain Test Suites: Regularly update and maintain automated test suites to ensure they remain relevant and effective as the application evolves.

Train the Team: Ensure that the team is well-trained in writing and maintaining automated tests. This includes understanding the tools, frameworks, and best practices for automated testing.

Commitment to Automated Tests = ensuring compliance with regulatory requirements –> all while supporting the flexibility and responsiveness that Agile methodologies promote.

GAMP®5 Categories
The process of assessing system components applies the GAMP software categories and hardware categories as input to establishing the required activities, based on how the system is constructed or configured.

The updated appendix in GAMP 5 emphasizes several key changes to categorization:

Component Integration: Computerized systems are typically composed of various components that span different categories, which should be considered as a continuous spectrum rather than discrete groups.

Risk-Based Scaling: The categorization of software is one aspect of a broader risk-based approach where the scale of life cycle activities is determined by the system's overall impact on good practices (GxP), its complexity, and its novelty.

This scaling is influenced by how critical the business process supported by the system is.

Software Categories: Despite a broader focus, software categories help determine the necessary rigor in supplier assessments and testing.

The ISPE GAMP®5 guide categorizes software into different types, each with specific considerations for validation based on their role and impact.

Each category requires a different level of scrutiny and validation based on the potential risk to GxP processes.

Category 1 includes basic software like operating systems, middleware, and tools for network monitoring and security in IT services. While generally reliable and indirectly tested through application testing, critical tools like those for password management should undergo specific risk assessments to decide if additional controls are needed.

(There is no Category 2)

Category 3 – Standard

These are off-the-shelf components that may require minimal configuration. Their validation depends on the extent of their configuration and their impact on GxP processes.

Category 4 – Configured

This category involves software that can be heavily customized to fit specific business processes. The validation process is more rigorous here due to the potential risks introduced by custom configurations. Supplier assessments and thorough testing of the configured application are critical to ensure functionality and compliance.

Category 5 – Custom

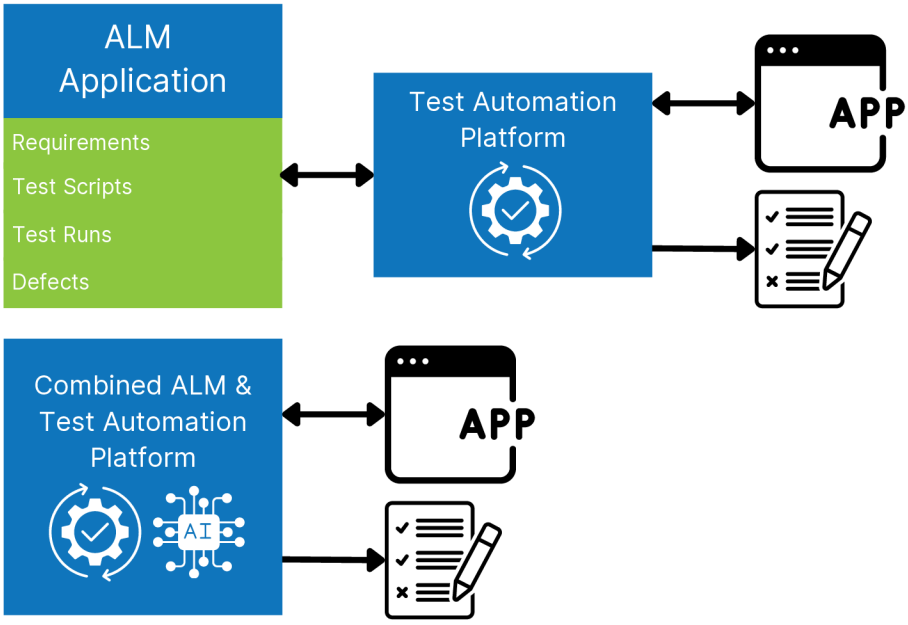
These are tailor-made solutions designed to meet specific needs of a regulated company. They carry a high risk due to the lack of prior user experience and potential for undetected errors. Rigorous functional risk assessments and validation are crucial here.

Chapter 3
Automated Computer Software Assurance (ACSA)

Modern ACSA Implementation

Automated Computer Software Assurance (ACSA) represents the next evolution in ensuring system compliance and reliability through automation. This chapter explores the practical implementation of ACSA, providing technical details and a step-by-step guide to integrating an ALM system with a Selenium platform.

The diagram below illustrates a modern ACSA implementation, highlighting the integration points between an Application Lifecycle Management (ALM) system and a Selenium-based automated testing framework. This model is used by many of the top life science manufacturers today. Key components include requirements management, test case creation, test execution, and result reporting. As AI testing automation matures, testing models will shift to a fully integrated solution which has all requirement, reporting, and testing functionality in one place.



What It Takes to Implement ACSA

- Requirements Management:
  - Define and document requirements within the ALM system.
  - Ensure traceability from requirements to test cases.
- Test Case Development:
  - Create automated test cases using Selenium.
  - Ensure test cases cover all critical requirements and regulatory needs.
- Integration Setup:
  - Establish integration between the ALM system and Selenium.
  - Configure tools for seamless communication and data exchange.
- Test Execution:
  - Execute automated tests via Selenium.
  - Monitor and log test results within the ALM system.
- Reporting and Documentation:
  - Generate detailed test reports.
  - Maintain documentation for regulatory compliance and audit purposes.

Step-by-Step Guide on Integrating an ALM System with a Selenium Platform

This guide provides a concise yet comprehensive framework, starting with environment preparation and progressing through defining requirements, developing test cases, configuring integration, executing tests, and reviewing results.
By establishing secure communication between the ALM system and Selenium, mapping requirements to test scripts, and automating test execution and result logging, organizations can enhance compliance, ensure thorough validation, and streamline their testing processes.

Step 1: Prepare the Environment
- Install and configure the ALM system.
- Set up Selenium and required libraries.
- Ensure both systems can communicate securely.

Step 2: Define Requirements in the ALM System
- Input system requirements into the ALM tool.
- Establish a hierarchy and relationships between requirements.

Step 3: Develop Automated Test Cases
- Write Selenium test scripts to validate requirements.
- Store test scripts in a version-controlled repository.

Step 4: Configure Integration
- Use APIs or middleware to connect the ALM system with Selenium.
- Map requirements in the ALM system to corresponding Selenium test cases.

Step 5: Execute Tests and Record Results
- Run Selenium tests from within the ALM system or through an integrated interface.
- Capture and store test results in the ALM tool.

Step 6: Review and Report
- Generate reports from the ALM system detailing test execution and results.
- Ensure reports include traceability from requirements to test outcomes.

Technical Architecture - Overview of the Technical Stack
- ALM System: Manages requirements, test cases, and traceability.
- Selenium: Automates web application testing.
- Integration Layer: Facilitates communication between the ALM system and Selenium (e.g., custom scripts, APIs).

Integration Points Between ALM and Selenium:
1. Requirements Linking:
   - Establish direct links between ALM requirements and Selenium test scripts.
2. Test Execution Management:
   - Trigger Selenium tests from the ALM system.
   - Capture execution status and results automatically.
3. Result Logging and Reporting:
   - Log test results back into the ALM system.
   - Generate compliance reports showing test coverage and outcomes.

Conclusion:
Implementing ACSA requires a detailed understanding of both the technical components and the integration points between the ALM system and Selenium. By following the outlined steps and leveraging the provided architecture, organizations can achieve efficient and compliant automated testing processes.

In the next chapter, we will delve into implementation of a regulatory case study using a CMMS/EAM application. This will demonstrate the practical benefits and outcomes of ACSA implementation.

Triggering a Test Case from ALM and Executing in Selenium with Jenkins

Integrating an ALM system with Selenium using Jenkins allows you to trigger test cases from the ALM, execute them in Selenium, and capture the results. Here's a detailed guide on how to set this up.

Prerequisites

- ALM System (e.g., ALM, JIRA, Azure DevOps)
- Jenkins installed and configured
- Selenium WebDriver and test scripts
- Source code repository (e.g., GitHub, Bitbucket)

1. Setup in ALM System

Define Test Cases and Requirements:

- Create and document test cases in your ALM system.
- Ensure each test case has a unique identifier (e.g., Test ID).

Configure Webhooks or API Triggers:

- Configure the ALM system to send triggers (e.g., via webhooks or REST API) to Jenkins when a test case needs to be executed.

Example Configuration for JIRA:

- Install and configure a Jenkins plugin for JIRA.
- Set up a webhook in JIRA to trigger Jenkins jobs based on specific events (e.g., issue transition to a specific state).

2. Setup Jenkins

Install Necessary Plugins:

- Install plugins for ALM integration (e.g., JIRA Plugin), Git Plugin, and Selenium Plugin in Jenkins.

Create a Jenkins Pipeline Job:

- From the Jenkins dashboard, create a new Pipeline job.
- Configure the pipeline script to handle test execution.

Pipeline Script Example - Appendix A

3. Execute Test Case from ALM

Trigger Jenkins Job from ALM:

- When a test case is ready to be executed, the ALM system triggers the Jenkins job, passing the test case ID as a parameter.

Example using JIRA:

- Transition the JIRA issue to a state that triggers the webhook.
- The webhook sends a request to Jenkins with the test case ID.

Webhook Example - See Appendix B

Triggering a Test Case from ALM and Executing in Selenium with Jenkins

4. Jenkins Executes Selenium Tests

Test Execution in Jenkins:

- Jenkins receives the trigger with the test case ID.
- Jenkins checks out the latest code from the repository.
- Jenkins sets up the environment and runs the specified Selenium test case using Maven or any other build tool.

Maven Command Example:

- The -DtestId=${params.TEST_ID} parameter is passed to the Maven command, which filters the tests to execute based on the provided test ID.

5. Capture and Report Test Results

Publish Test Results in Jenkins:

- Jenkins publishes the test results using JUnit or TestNG plugins.
- Test results are stored and displayed in Jenkins for review.

Update ALM System with Results:

- Configure post-build actions in Jenkins to update the ALM system with test results.
- Use Jenkins plugins or custom scripts to send the test results back to the ALM system.

Conclusion

By integrating Jenkins with an ALM system and Selenium, you can create a seamless pipeline where test cases are triggered from the ALM, executed in Selenium, and results are automatically captured and reported back to the ALM.

This setup enhances efficiency, ensures continuous compliance, and provides quick feedback on test outcomes. The next chapter will focus on a regulatory case study, demonstrating the practical benefits and outcomes of ACSA implementation.

Chapter 4 - Regulatory Compliance and Technical Case Study for a CMMS System

Functional Aspects of a CMMS System

A CMMS typically manages maintenance activities, schedules, and records. Key functionalities include:

- **Work Order Management:** Creation, assignment, and tracking of maintenance tasks.
- **Asset Management:** Tracking and managing physical assets, including their maintenance history.
- **Preventive Maintenance Scheduling:** Automating maintenance schedules based on time or usage triggers.
- **Inventory Management:** Managing spare parts and inventory levels.
- **Reporting and Analysis:** Generating reports for compliance, performance analysis, and decision-making.

These functionalities directly relate to compliance by ensuring that maintenance activities are documented, tracked, and executed according to regulatory standards.

Regulatory Requirements - 21 CFR Part 11 Requirements Applicable to a CMMS System

21 CFR Part 11 outlines the criteria under which electronic records and electronic signatures are considered trustworthy, reliable, and equivalent to paper records and handwritten signatures. For a Computerized Maintenance Management System (CMMS), the following requirements are particularly relevant:

- **Validation**: Ensure that the CMMS can perform its intended functions consistently and accurately.
- **Audit Trails**: Implement secure, time-stamped audit trails to record the actions taken in the system.
- **Security Controls**: Establish strict access controls to prevent unauthorized access and data manipulation.
- **Electronic Signatures**: Ensure electronic signatures are unique to an individual and cannot be reused.
- **System Documentation**: Maintain detailed documentation of system configurations, validations, and usage.

Mapping Requirements to Automated Test Cases

The table below maps key 21 CFR Part 11 requirements to corresponding automated test cases:

Regulatory Requirement	CMMS Functionality	Automated Test Case Description
Validation	Work Order Management	Verify that work orders can be created, assigned, and tracked accurately.
Audit Trails	Asset Management	Ensure audit trails are generated for asset updates.
Security Controls	User Access Management	Test that unauthorized users cannot access or modify the system.
Electronic Signatures	Work Order Approval	Validate that electronic signatures are required and verified for work order approvals.
System Documentation	Reporting and Analysis	Check that system configurations and usage reports are accurately generated and stored.

Technical Walkthrough - Creating a Simple Test Suite for a Sample CMMS Application

Step-by-Step Instructions and Code Example

1. Set Up the Environment
   - Install Selenium WebDriver.
   - Set up a CMMS application (use a demo or sample application).
2. Write Test Cases
3. Execute Tests and Capture Results
4. Set up the test suite in Jenkins.
5. Configure Jenkins to execute the Selenium tests and capture results.
6. Review and Report
7. Generate and review test execution reports in Jenkins.
8. Ensure results are logged and mapped to corresponding requirements.
9. Provide Screenshots to Guide Readers
10. Insert Screenshots Here
11. Screenshot Descriptions:
12. Login Page of the CMMS application.
13. Work Order Creation Form.
14. Success Message after Work Order creation.
15. Jenkins Job Configuration for test execution.
16. Test Execution Results in Jenkins.

Implementing ACSA with a CMMS system ensures compliance with 21 CFR Part 11 by automating validation processes and enhancing system reliability.

By following the steps outlined in this chapter, organizations can achieve significant time and cost savings, improve test accuracy, and streamline their compliance efforts.

This hands-on project provides a practical example of setting up and executing automated tests, demonstrating the tangible benefits of an integrated requirements management and automated testing system.

Example Test Case 1: Verify Work Order Creation

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class CMMSWorkOrderTest {
    public static void main(String[] args) {
        // Set up WebDriver
        System.setProperty("webdriver.chrome.driver",
"path/to/chromedriver");
        WebDriver driver = new ChromeDriver();

        // Navigate to CMMS login page
        driver.get("http://cmms-demo-url/login");

        // Log in to CMMS
        WebElement username = driver.findElement(By.id("username"));
        WebElement password = driver.findElement(By.id("password"));
        WebElement loginButton = driver.findElement(By.id("loginButton"));
        username.sendKeys("admin");
        password.sendKeys("admin123");
        loginButton.click();

        // Navigate to Work Order creation page
        driver.findElement(By.id("createWorkOrder")).click();

        // Fill out Work Order form
        driver.findElement(By.id("workOrderTitle")).sendKeys("Test Work
Order");
        driver.findElement(By.id("workOrderDescription")).sendKeys("This is
a test work order.");
        driver.findElement(By.id("submitWorkOrder")).click();

        // Verify Work Order creation
        String successMessage =
driver.findElement(By.id("successMessage")).getText();
        if(successMessage.contains("Work Order created successfully")) {
            System.out.println("Test Passed: Work Order created
successfully.");
        } else {
            System.out.println("Test Failed: Work Order creation failed.");
        }

        // Close the browser
        driver.quit();
    }
}
```

Summary

This document has guided you through the essential steps of implementing Automated Computer Software Assurance (ACSA) using Selenium integrated with an ALM solution.

We touched upon:

- The value of Automated Testing by explored how automated testing can save time and costs, increase test coverage, and improve accuracy. Real-world case studies highlighted the substantial benefits, demonstrating reduced manual testing hours and enhanced compliance.

- CSV and CSA Key concepts of Computer System Validation (CSV) and Computer Software Assurance (CSA) were introduced, along with their risk-based approaches. We also discussed recent updates to the GAMP5 model and relevant regulatory requirements.

- Automated Computer Software Assurance (ACSA) Practical implementation details of ACSA were provided, including integrating an ALM system with Selenium. This section covered everything from setting up the environment to executing tests and logging results, emphasizing the technical stack and integration points.

- In the Regulatory Compliance and CMMS Case Study, we identified 21 CFR Part 11 requirements for a CMMS system and mapped these to automated test cases. The benefits of a fully integrated requirements management and automated testing system were discussed, including significant time and cost savings. A hands-on project guided you through creating a simple test suite for a CMMS application.

Investing in modern CSV practices is an investment in your organization's future, ensuring you stay ahead in a competitive and regulated industry.

Ready to advance your automated testing?

Book a call with our experts to discuss how we can help you implement ACSA and achieve your compliance goals.

Book a Call: [Link to Schedule a Call]

Appendix A

```
pipeline {
    agent any
    parameters {
        string(name: 'TEST_ID', defaultValue: '', description: 'The ID of the test case to execute')
    }
    stages {
        stage('Checkout') {
            steps {
                git url: 'https://github.com/your-repo.git'
            }
        }
        stage('Setup Environment') {
            steps {
                sh 'source setup_environment.sh'
            }
        }
        stage('Run Tests') {
            steps {
                sh 'mvn clean test -DtestId=${params.TEST_ID}'
            }
        }
    }
    post {
        always {
            junit 'target/surefire-reports/*.xml'
        }
        success {
            script {
                // Update ALM system with test results here
                // e.g., update JIRA issues or HP ALM
            }
            emailext (
                subject: "Test Successful: ${params.TEST_ID}",
                body: "The test case with ID ${params.TEST_ID} has passed.",
                recipientProviders: [[$class: 'DevelopersRecipientProvider']]
            )
        }
        failure {
            emailext (
                subject: "Test Failed: ${params.TEST_ID}",
                body: "The test case with ID ${params.TEST_ID} has failed. Please check the Jenkins console output for details.",
                recipientProviders: [[$class: 'DevelopersRecipientProvider']]
            )
        }
    }
}
```

Appendix B

```
{
 "name": "Trigger Jenkins Job",
 "url": "http://your-jenkins-url/job/your-pipeline-job/buildWithParameters?TEST_ID={{issue.key}}",
 "events": [
   "jira:issue_updated"
 ],
 "jqlFilter": "project = YOUR_PROJECT AND status = Ready for Testing",
 "excludeBody": false
}
```