

Ready, Set, Automate - Best Practices in Using Automated Tools for Validation

Eli Miller¹ and Gayathri Kolandaivelu²

¹Covance Laboratories Inc., Madison, WI; ²Janssen Research & Development, Integrated Data Analytics & Reporting (IDAR), Spring House, PA

Introduction

Patients and regulators have an expectation that clinical data is accurate, and its analysis is sound. A PhUSE working group was formed to discuss and determine best practices for Quality Control and Validation. This poster will focus on the general function and benefit of quality control tools, investigate how QC tools can enhance the validation process and workflow and explore future and cutting-edge considerations for testing and validation of software. The tools elaborated are Continuous Integration and Delivery, Automated Structural testing, automated functional testing, and Code Coverage Analysis. There are countless software and services that seek to streamline the clinical software development process, therefore this poster will focus on the general function, best practices, use cases and benefit of these services.

Automated tools support for quality control and quality assurance of the processes used in clinical programming. Repetitive tasks can seamlessly be replaced by robust automated tools.

Cutting Edge Considerations

With new technologies come new approaches. The increase of processor speed and decrease of storage cost has opened the door for less traditional methods for analysis and automation. As an organization begins to use more sophisticated methods to determine a compounds effect, the process and tools used to ensure quality of software must also become more sophisticated. Creating and testing custom analysis software is becoming more common for research organizations. Testing complex analysis software and applications can be assisted with tools that test frameworks and the functionality of an application. Using a novel application can carry an inherent risk. Understanding the limitations of a program is necessary for an organization and the tools that can assess limitations are mature and accessible.

Functional Testing

Functional testing refers to testing output and performance of an application without taking the underlying structure of the code into consideration. It is focused on the function of the application from the perspective of the end user. Functional testing often involves testing different valid, invalid, and boundary inputs and ensure the output is as expected. These tests are usually user use cases and generally don't involve the internals of the program. These can be automated, using automated browsers or JavaScript testing, if it's a web browser-based application, however most of these are user tests.

Structural Testing

Structural testing refers to testing the way a program is designed and written. This can be as simple as a code review or as complex as a system of complete path and unit testing. This doesn't always need to test the overall function of the code, but instead ensures the code is written to design specifications and uses efficient data structures. Code review is a systematic process for ensuring the software code is written based on organizational and project guidelines. Code coverage is the range of possible states that code is prepared to handle. Path testing is a structural testing method that involves using the source code of a program in order to find every possible executable path. Unit testing is the process of testing the functional components of software and ensuring they are performing as expected.

Nearly all programming languages have unit testing frameworks. JUnit(Java), unittest(Python), testthat(R), are a few popular unit testing frameworks that can test the foundation of your program and verify they are behaving as expected.

Code Coverage

Code coverage is a measurement of how many lines, statements, or blocks of your code are tested using your suite of automated tests. It's an essential metric to understand the quality of your QA efforts. Code coverage shows you how much of your application is not covered by automated tests and is therefore vulnerable to defects. It verify assumptions in a code and the code that is accounted for testing.

Code coverage is a white box testing methodology, that is it requires knowledge of and access to the code itself rather than simply using the interface provided. Code coverage is probably most useful during the module testing phase, though it also has benefit during integration testing and probably at other times, depending on how and what you are testing. A number of different metrics are used determine how well exercised the code is. A few common ones are statement coverage, branch coverage, path coverage, condition coverage and time coverage.

Related Links
QC and Validation Working Group:
https://www.phusewiki.org/wiki/index.php?title=Best_Practices_for_Quality_Control_and_Validation

References
Since this poster is a portion of GPP white paper, references may or may not be limited to the scope of the poster. A comprehensive reference list will be provided upon request.

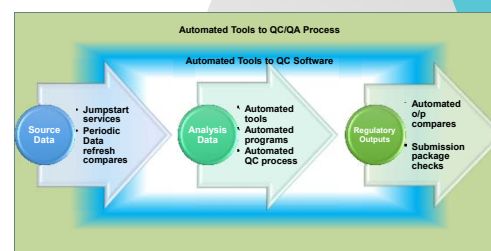


Figure 1. Basic components of clinical reporting.

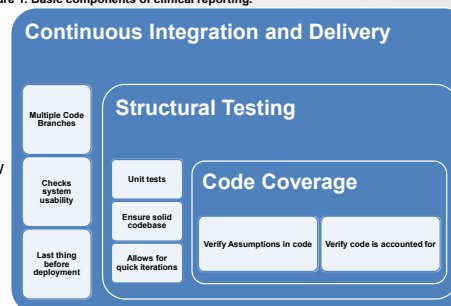


Figure 2. Continuous integration and delivery services.

Continuous Integration and Delivery

Using an iterative approach to development can result in continuous improvement to your statistical software. However, as applications get larger, the ability for unintended effects of enhancements can grow. The ability to test a program while it is being built can improve quality and speed of a development team. This is an essential part of an iterative development process. These services will build and test a program as it is improved upon.

There are several Continuous Integration services (Figure 2) that can interface with your code repository that will test and build your application before you release it into production. This can reduce blind spots and ensure your tests are being run each time.

Conclusion

These tools, when used thoughtfully, can coordinate teams of programmers, and ensure that software is robust and validated. In this technology era we live in, Automation tools are like our programming partners to reduce human errors and improve quality across the statistical process. This poster is a portion of the Quality control and validation white paper which is being developed by several industry members.