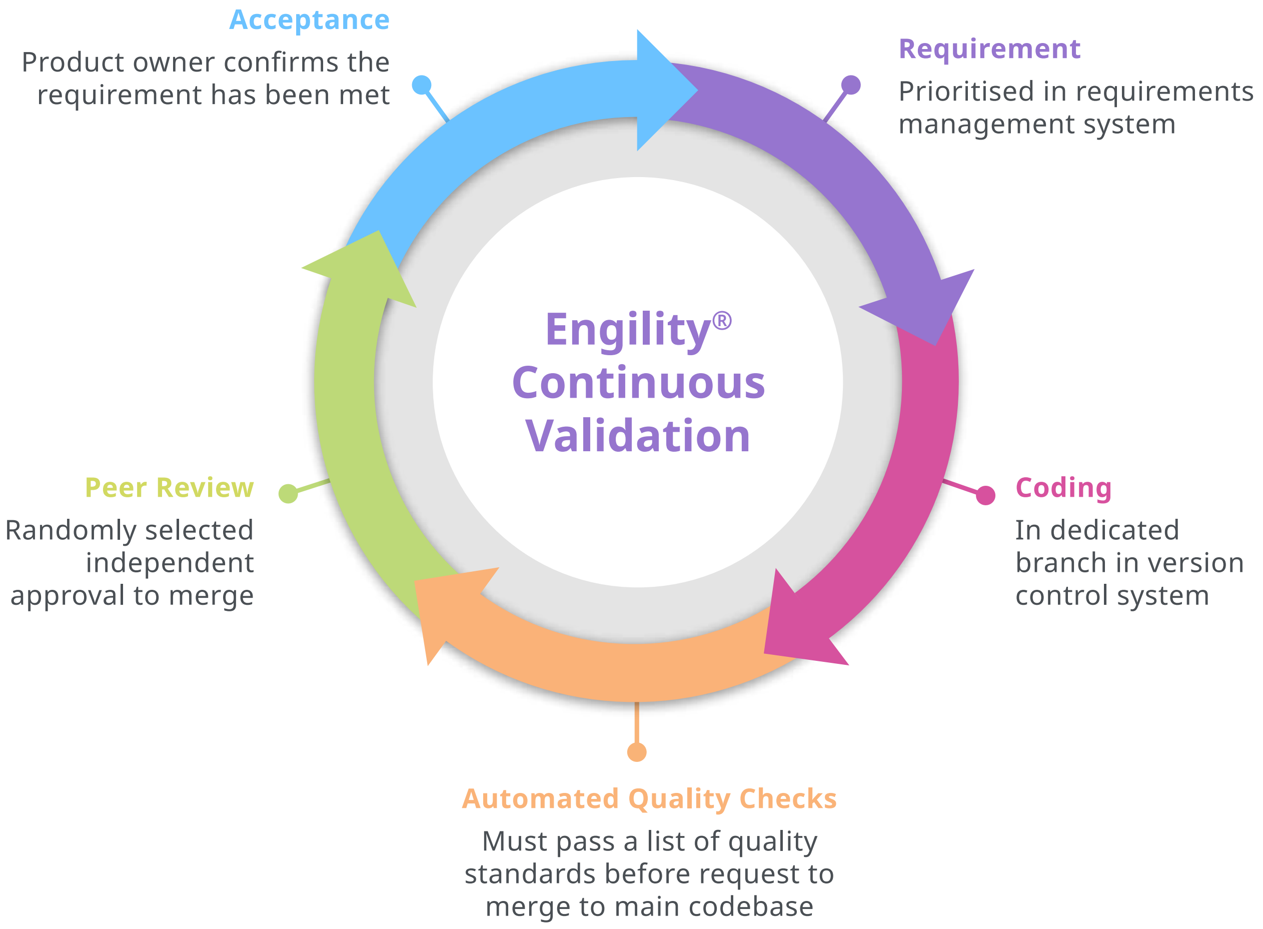
White Paper: 16 May 2024

Has validation in an agile software development environment come of age?



Acceptance
Product owner confirms the requirement has been met
Requirement
Prioritised in requirements management system
Engility® Continuous Validation
Coding
In dedicated branch in version control system
Automated Quality Checks
Must pass a list of quality standards before request to merge to main codebase
Peer Review
Randomly selected independent approval to merge

Figure 1. The Continuous Validation Cycle.

PHARMASEAL
Trials transformed

Introduction

Although Agile software development has its origins in the 70s (and probably earlier), I remember it becoming a topic of interest in Pharma IT circles in the late 1990s / early 2000s - around the time that the Agile Manifesto[1] was published. The concept of flexibility and quick feedback was very exciting and made perfect sense, but the lack of formality and documentation made it seem incompatible with regulated environments.

Over time it has become accepted that agile software development can work well within a formal project and today most software vendors supplying the pharmaceutical industry have an agile aspect to their Software Development Life Cycle (SDLC). Most of the approaches that I have seen as a customer have been a hybrid of traditional validation approaches coupled with an agile software development process. This allows the development team to get rapid feedback from the product managers but still requires a lengthy validation phase when a new release is delivered to the customer.

At PHARMASEAL, we have developed a process, that we called Continuous Validation, that has the quick feedback loop and course correction aspects that makes agile so attractive but also maintains, in real-time, the control and traceability that is required by validation. So we can react quickly to changing requirements and deliver quality software to the customer both quickly and efficiently. Recent blog posts show that PHARMASEAL is not alone and that other companies are using similar Continuous Validation SDLCs. Furthermore, the principles of Continuous Delivery, upon which these processes are based, are becoming established in other regulated environments, such as banking, and have been shown to be well suited to these environments.

In this article I will describe two examples of Continuous Validation SDLCs - ours and TrialGrid's - and look at the aspects of Continuous Delivery that make it particularly suitable for validated applications used in regulated industries.

Software Development Life Cycle

The PHARMASEAL SDLC consists of a continuous cycle of discrete steps, with 1 iteration for each feature implemented; see figure 1 for a summary. Each step is described in more detail below.

Requirements are captured in our requirements management tool. All work on our software - features, bugs and housekeeping - go into this tool as small chunks of work. We can re-order these chunks of work into a prioritised list and interweave release markers to manage application development.

All the components of our application are **coded** and managed in a version control tool: the source code, test code, build scripts, etc. - everything you need to test and build the system. The tool maintains a complete history of the application and allows you to roll back to any previous state.

Automated testing - each new feature added to the system has a set of automated tests. These form our regression test suite - which is run for every change to the application. Each test is tagged with the requirement number to maintain traceability. In addition to the testing, we run automated checks to scan the source code for security vulnerabilities and to confirm the code meets our coding standards.

After the automated tests, a mandatory independent **Peer Review** ensures that the change meets our quality requirements for design, implementation and testing.

The final step is **acceptance** where the Product Owner confirms that the change satisfies the requirement. If any changes are required, the developer must submit them through a repeat of the above process.

The Continuous Validation cycle is the image at the first page of this white paper.

The whole cycle is managed with an integrated set of tools, all with user authentication and audit trails, so every activity is recorded against who did it and when. This gives us a complete history of the development of the application and also enables us to extract the data to create documentation such as user requirements specification, functional specification, test reports and traceability matrix.

Once a change has been committed and accepted, it is considered done and ready for deployment but is held until the next planned release. During the release process (see figure 2) the accepted source code is branched from QA to Release Candidate to freeze it while development continues. The Release Candidate then goes through the validation process, after which it can be promoted to Production. The release process can comfortably be completed in a day but is normally held back from production for 30 days to allow time for customer review.

Software Development Life Cycle cont.d

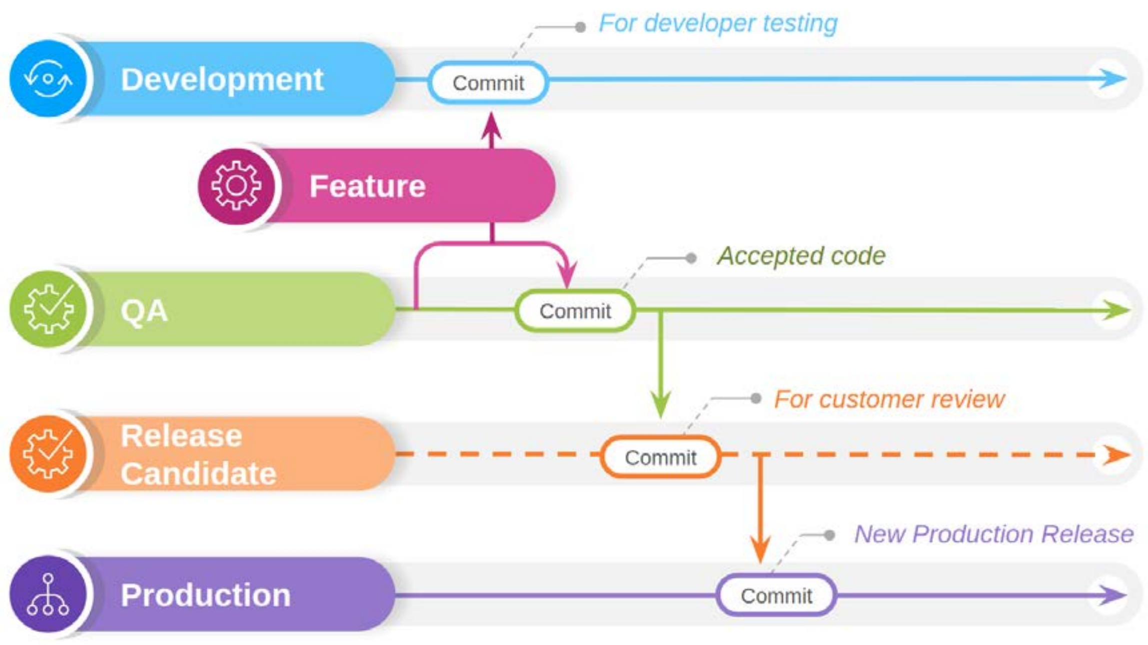


Figure 2. The PHARMASEAL release process.

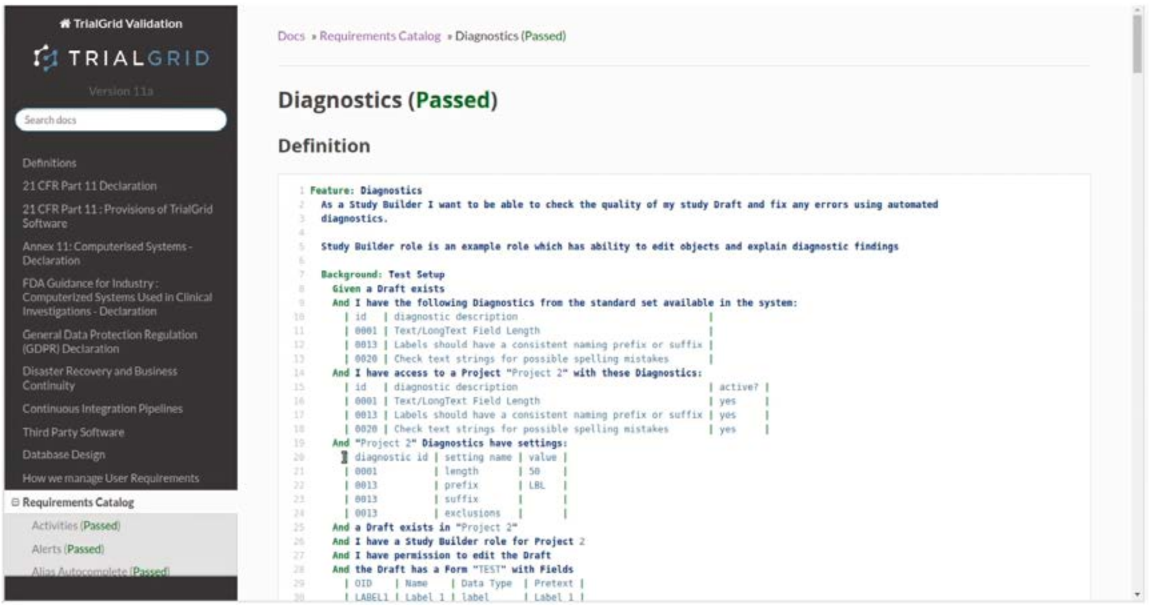
Continuous Validation Example 2

TrialGrid

TrialGrid have described a similar Continuous Validation process in recent blogs[3,4]. They center their process around their version control tool which maintains all their application assets and manages their continuous validation pipeline. The requirements (and bugs) are tracked in an Issue list with unique identifiers that are referenced in the commit messages, and tagged in the testing, to enable traceability.

Every component is under version control and as each change is committed, the application is automatically built in the Dev environment for review. An automated pipeline of around 40 tests and other quality checks are run and all of them must pass for the process to continue. A peer review is required for the change to be accepted and this process is automatically recorded by the version control system with comments and responses tracked along with who did what and when.

The commit process also includes creation of an integrated hyperlinked validation package - generated automatically from the artefacts, workflow and audit trail data held in the system. This package organises the validation evidence so that an audit can be remote and largely self-guided (see figure 3).


TrialGrid Validation
TRIALGRID
Version 11a
Search docs
Definitions
21 CFR Part 11 Declaration
21 CFR Part 11 : Provisions of TrialGrid Software
Annex 11: Computerised Systems - Declaration
FDA Guidance for Industry : Computerized Systems Used in Clinical Investigations - Declaration
General Data Protection Regulation (GDPR) Declaration
Disaster Recovery and Business Continuity
Continuous Integration Pipelines
Third Party Software
Database Design
How we manage User Requirements
Requirements Catalog
Activities (Passed)
Alerts (Passed)
Alias Autocomplete (Passed)
Docs » Requirements Catalog » Diagnostics (Passed)

Diagnostics (Passed)

Definition

```
Feature: Diagnostics
  As a Study Builder I want to be able to check the quality of my study Draft and fix any errors using automated
  diagnostics.

  Study Builder role is an example role which has ability to edit objects and explain diagnostic findings

  Background: Test Setup
    Given a Draft exists
    And I have the following Diagnostics from the standard set available in the system:
      | id   | diagnostic description                             |
      | 0001 | Text/LongText Field Length                         |
      | 0013 | Labels should have a consistent naming prefix or suffix |
      | 0020 | Check text strings for possible spelling mistakes  |
    And I have access to a Project "Project 2" with these Diagnostics:
      | id   | diagnostic description                             | active? |
      | 0001 | Text/LongText Field Length                         | yes     |
      | 0013 | Labels should have a consistent naming prefix or suffix | yes     |
      | 0020 | Check text strings for possible spelling mistakes  | yes     |
    And "Project 2" Diagnostics have settings:
      | diagnostic id | setting name | value |
      | 0001          | length       | 50    |
      | 0013          | prefix       | LBL   |
      | 0013          | suffix       |       |
      | 0013          | exclusions   |       |
    And a Draft exists in "Project 2"
    And I have a Study Builder role for Project 2
    And I have permission to edit the Draft
    And the Draft has a Form "TEST" with Fields
      | OID    | Name    | Data Type | Pretext |
      | LABEL1 | Label 1 | label     | Label 1 |
```

Figure 3. A screenshot of the TrialGrid validation package..

PHARMASEAL
Trials transformed

Continuous Validation Example 2

TrialGrid cont.d

Each successful change is delivered immediately to the Beta environment. Periodically, at times agreed with their customers, the accumulated changes are promoted to Production. For production releases, there is an additional manual review and the validation package is signed and retained.

TrialGrid has been running this process, with periodic refinement, successfully for more than 2 years, and it has been reviewed by a number of customer audits.

Continuous Delivery = Continuous Compliance

Both the PHARMASEAL and TrialGrid processes are based on the principles of Continuous Delivery. Dave Farley, author and speaker, has recently posted a blog[5] describing how Continuous Delivery (CD) satisfies the key requirements of regulatory compliance.

The features of Continuous Delivery that make it ideal for a validated environment are:

Everything is kept under version control - the source code, database build scripts, test code, system build scripts - everything that you need to create the system. Thus it is possible to see what state the application was in at any point of time, and every action can be traced to who and when, and often why, it was carried out.

Automation of repetitive tasks - in particular testing, code quality checks, build and deployment - making them fast, easy, reproducible and free from human error.

Continuous integration and continuous testing - a repeatable reliable development process with incremental change and quick feedback.

Build quality in - automation of quality checks means issues are picked up early and fixed before they become part of the application.

> **When describing the Continuous Delivery approach to people I am regularly asked, “Yes,that all sounds very good, but it can’t possibly work in a regulated environment can it?” I have come to the opposite conclusion. I believe that CD is an essential component of ANY regulated approach.**
>
> **Dave Farley**

PHARMASEAL
Trials transformed

Continuous Delivery has also recently been shown scientifically to be linked to higher software delivery performance[6]. More significantly - from a validation perspective - it has been shown to be predictive of specific measures of quality - lower change fail rates and lower levels of rework or unplanned work (time spent on break-fix, emergency software deployments and patches) - the latter in a statistically significant way.
Thus there is good evidence that Continuous Delivery does lead to higher quality software development.

Summary

Software in the pharmaceutical industry has always been conservatively managed so it is not surprising that Agile methodologies would be slowly and gradually adopted, especially when the Agile Manifesto plays down the value of documentation and process which are the core of validation. The significance of the emergence of Continuous Delivery is that it emphasises technical practices which bring greater capability and control. These practices do not conflict with the more ceremonial team and management practices of other agile methodologies. Indeed they are complementary and we have implemented the technical practices of CD along with sprints, stand-ups, and retrospectives taken from Scrum[7].

The principles of CD, together with the availability of easily-implemented, powerful tools to support it, provide a secure framework that enables truly agile software delivery whilst genuinely enhancing quality and control. CD promotes fast feedback - while the work is still fresh in everyone’s mind - so fixes and course alterations are quicker and easier. The traditional methods of testing and carrying out security reviews at the end of a development cycle puts these steps on the critical path - every fix required slows down delivery and, to compound things, the issues have been given time to become embedded, making them more difficult to fix. Another big benefit of the continuous validation approach is that every change - even an emergency hotfix that needs to be deployed to production that night - is of the same quality. If you rely on manual testing and documentation you are forced to make compromises in these situations.

I have no doubt that we will see more examples of SDLCs featuring these processes in the near future, and customers, suppliers and the regulatory authorities will all benefit from the increased efficiency and control that they provide. It will take some time for vendors with established products to make the transition - as the changes are fundamental - but the benefits are clear and the decision will be more about how and when rather than why. I do think that we can safely say that Agile software development in a validated environment has definitely come of age.

References

1. Agile Manifesto; *http://agilemanifesto.org/*
2. Why Continuous Validation is important in Agile Product Development - PHARMASEAL Blog; *https://www.pharmaseal.co/news/why-continuous-validation-is-important-in-agile-product-development/* 29th April 2019
3. Continuous Validation (2019 Edition) - TrialGrid Blog; 1st May 2019; *https://www.trialgrid.com/blog/continuous_validation_2019.html*
4. Tracing Bug Fixes to the code - TrialGrid Blog: 23rd September 2019; *https://www.trialgrid.com/blog/traceability_to_tests.html*
5. Continuous Compliance - Dave Farley's Weblog; 3rd September 2019; *http://www.davefarley.net/?p=285*
6. Accelerate: the science behind DevOps: building and scaling high performing technology organizations; Nicole Forsgren, PhD., Jez Humbler and Gene Kim, ISBN 9781942788331
7. Scrum.org; *https://www.scrum.org/*

Author

Hugh O'Neill is Vice President, Quality and Operations at PHARMASEAL International Ltd. Hugh is a PhD scientist with more than 30 years experience in the pharmaceutical industry, much of it with IT responsibility.

To speak with an expert and get more information please visit **pharmaseal.co**

PHARMASEAL
Trials transformed