Thomas Lüftl

Lechbrucker Straße 31
81476 München
E-Mail: Thomas.@Lueftl.com
Telefon: +49 89 74654898
Mobil: +49 172 68 36 030
Fax: +49 3212 1298171
www.lueftl.com

Agile Implementation of LIMS Software Reduce Validation or Verification Effort and Meet GAMP 5 Standards

1. Problem Statement

Formal software verification (often called validation) is a necessity in regulated pharma environments, as for example described and recommended in the GAMP 5 standard. Verification is seen as a heavy burden in the implementation of Laboratory Information and Management Systems (LIMS). To help out their clients, LIMS vendors offer validation support mainly for “OQ” (OQ means “Operational Qualification”) for their clients (hereinafter called “users”). LIMS vendors provide their users with scripts based on which the technical functionality of the software in the clients’ (users’) environment (database, security, operating system, hardware platform, etc.) is tested.

As often seen in practice, OQ may take up to 50 % of the total effort outsourced for a project. It is questionable whether such an effort is constructive for configurable software packages (GAMP 5 category 4). Vendors of software for regulated pharma environments maintain a state of the art quality assurance system and they have to prove that their core system is encoded and tested comprehensively.

2. Background

The GAMP 5 standard mainly refers to the traditional V-model or waterfall project management model that has been the preferred model for software development for a long period of time. The V-model means that all functional specifications and a complete configuration/system design are derived from the user requirements. The system is then built, tested against configuration specification, functional specification and finally against user requirements by an acceptance test.

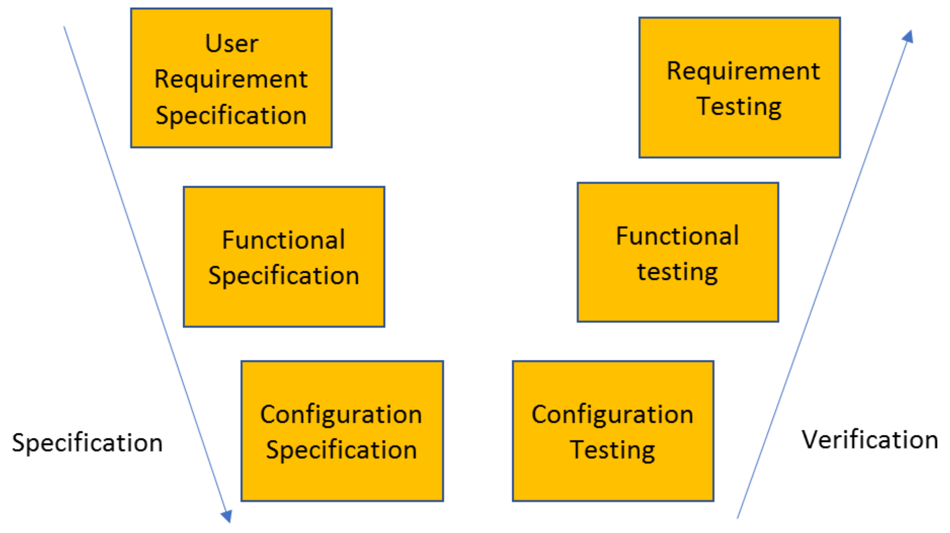

Figure 1: V-model or waterfall model

There are two disadvantages of the traditional V-model: There are long cycle times from user requirement specification to user acceptance test and requirements may change in the meantime. The modelling of user requirements without seeing a running piece of software is abstract and usually requirements for modification arise when the final user deals with the running software for the first time.

Latest approaches for software projects (like Rapid Prototyping or agile methods like Scrum) provide shorter cycle times and a more immediate look and feel of the software for the final user. The idea is that in a series of consecutive “sprints”, a working and potentially releasable increment of product (software) is completed every 2 - 4 weeks. During a sprint, design, development and testing people work closely together and produce a running increment of software that has already been tested. Incremental approaches are particularly valuable for packaged software (they already have a starting point) and for situations where final users experience difficulties expressing user requirements beforehand and in an abstract way.

3. Objective

Companies should look for ways to reduce software verification effort according to GAMP 5 recommendations.

4. Proposed Solution

Software verification according to GAMP-5 should be integrated in processes that have to be done anyhow with each increment (sprint) during software implementation. This is functional testing by the development team during the sprint and requirement testing by the product owner and key users at the end of each sprint. These two steps of quality assurance have to take place in any case. In a regulated environment, there may be some additional effort for documentation.

Supplier activities should be levered to the maximum possible extent and a supplier audit should be performed. Such an audit should include for example software/hardware design practices, product design records, program coding standards, system test records, programming tools, code review practices, version control, software replication, problem reporting/resolution and fault notification to customers. The user should focus on the process: is the software able to support the process and is there any risk that automation adds to the process? A mere retesting of the vendor’s software in the environment of the user is not constructive.

The software project for laboratory automation should be clearly structured by use cases. An increment or sprint should be a use case or a defined set of use cases. A project sequence by use case is due to the fact that LIMS systems have a specific nature of complexity. LIMS complexity is different from the complexity of ERP systems. ERP systems picture company processes end-to-end and cross-functional. They integrate processes of many different organizational units of the user, like purchasing, production, warehousing, HR, accounting, sales, order processing, invoicing, collection, etc. In an ERP system, complexity is driven by cross-functionality. In contrast to ERP, the scope of LIMS is usually reduced to a laboratory and some internal or external users of the analysis results. LIMS complexity is driven by the variety of use cases. Use cases are formed by different analytical methods, different calculations and different artifacts. The structure of use cases is often multidimensional. For a bioanalytical laboratory, such a structure could be for example:

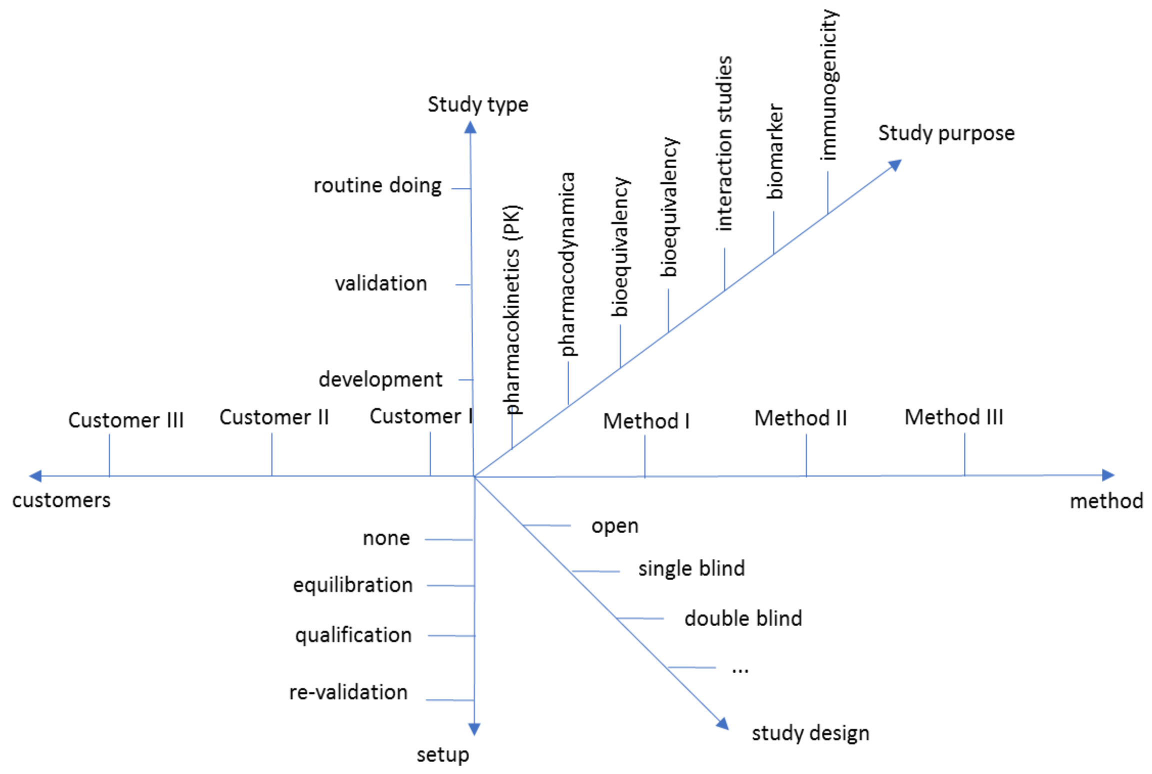

Figure 2: Dimensions of use cases

Dimensions in these examples are obviously different analytical methods; furthermore, also study purpose, study design, setup, customers, study type. Different study purposes mean different calculations and graphs (like PK). Some study types, like blind studies, require special sample handling and coding and processes when codes are opened at the end of a study. In longer running routine studies (like in clinical phases 2 - 4) samples arrive over a longer period of time and there is the necessity to set up instruments several times when new samples arrive. With each setup, some kind of verification has to be made and this may substantially differ in scope and complexity: full re-validation, qualification, equilibration. Different customers may require special artifacts. The distinction of cases as depicted applies for routine studies with a validated method. Method development and validation are important tasks of a laboratory to enable routine studies—and they create artifacts that differ from routine studies.

Increments or sprints in a software implementation project are blocks or slices in such a structure and each of them comprises one or more use cases. In our example, a use case would be: a single-blind PK study with one method and no specific customer requirements.

The flow diagram in Figure 3 shows the sequence of tasks in agile/incremental software configuration and implementation projects.

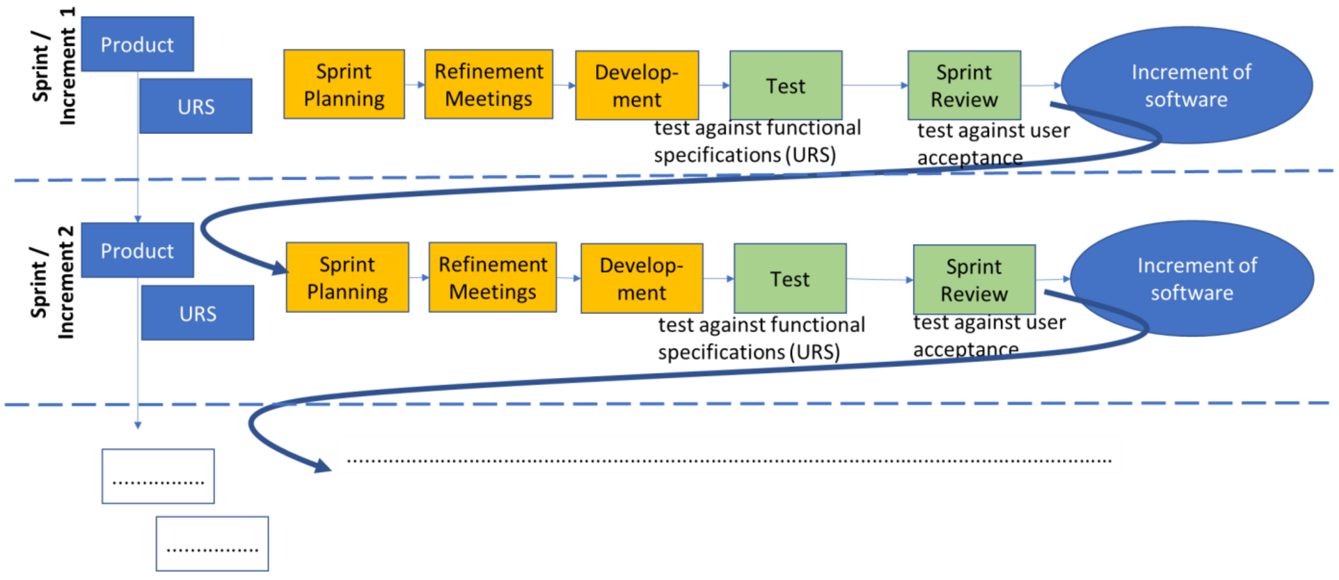

Figure 3: Flow of work in agile/incremental software configuration

A list of desired functionalities (so called product backlog) is maintained and updated throughout the configuration work as new insights are generated. User requirements are a refinement of the product backlog items. Tests (functional and user acceptance) are performed with each sprint/increment. Along the way, further regression/integration testing will be required.

Software verification and testing is constructive when it is done this way. “Validation” in a regulated environment may still be seen as a burden and nitpicking. Nevertheless, when organized well most of the verification work will be tasks that should be performed anyhow as quality assurance steps in an IT project.

5. Conclusion

Vendors of LIMS software offer standardized validation packages for “OQ” by means of automated and standardized methods.

Users should carefully reflect how much of this onsite testing of standard software is constructive and to which extent a vendor audit can help.

Verification at the user site should focus on specific user processes and be incremental. With each increment of software configuration, a verification of software against process and user requirements should be performed taking into account all different cases and process variants.

Such verification is constructive, as it assures the quality of the software implementation project and reduces the risk for the user and the patient.

Author

Thomas Lueftl
Lechbrucker Straße 31
81476 München
E-Mail: Thomas@Lueftl.com
F: +49 89 74654898
M: +49 172 68 36 030
F: +49 3212 1298171
Xing: https://www.xing.com/profile/Thomas_Lueftl2
LinkedIn: www.linkedin.com/in/thomaslueftl
www.lueftl.com