**BDD - A Practicable Approach for Computerised System Validation?**

**Bachelor Thesis 2020**

Client: wega Informatik AG  
Author: Sabrina Leuenberger  
Lecturer: Stephan Jüngling  
City, Date: Basel, 23th of July 2020

|  |  |
| --- | --- |
| **BDD – A Practicable Approach for Computerised System Validation?** |  |
| **Author** |  |
| Sabrina Leuenberger Merlachfeld 54 3280 Murten +41 (0)78 935 19 99 saleuenberger@gmx.ch |  |
| **Lecturer** |  |
| Stephan Jüngling University of Applied Sciences and Arts  Northwestern Switzerland  stephan.juengling@fhnw.ch |  |
| **Client** |  |
| wega Informatik AG Mathias Fuchs und Evelyne Daniel Aeschengraben 20 4051 Basel +41 (0)61 270 87 87 info@wega-it.com www.wega-it.com |  |
| Basel, July 2020 |  |

Declaration of Authenticity

I the undersigned declare that I have prepared the present paper independently and without the use of sources other than those indicated in the reference list.

All statements and information contained herein are listed and indicated as quotations and / or paraphrases.

This Bachelor Thesis / Project Work / Student Research has not been published to date. It has thus not been made available to other interested parties or examination boards.

City, Date

Signature

Preface or Background of the Project or Acknowledgment

Acknowledgements:

* wega Informatik AG for hosting the project, access to Teams, Confluence and Jira as collaboration tools
* Evelyne Daniel and Mathias Fuchs from wega for all the support and valuable inputs, their time, ….
* Andreas Hosbach from Zühlke AG for all is support from the technical side, ……
* Stephan Jüngling for supervising this project from FHNW,……
* Manuel Kohler: Jira und Confluence

Management Summary / Abstract

Table of Contents

[Declaration of Authenticity II](#_Toc44339628)

[Preface or Background of the Project or Acknowledgment III](#_Toc44339629)

[Management Summary / Abstract IV](#_Toc44339630)

[Table of Contents V](#_Toc44339631)

[1 Introduction 1](#_Toc44339632)

[1.1 Initial Situation 1](#_Toc44339633)

[1.2 BDD High Level Test Automation 1](#_Toc44339634)

[1.3 Automated Testing for OQ 2](#_Toc44339635)

[1.4 Hypothesis and Research Questions 2](#_Toc44339636)

[1.5 Scope 3](#_Toc44339637)

[1.5.1 In Scope 3](#_Toc44339638)

[1.5.2 Out of Scope 3](#_Toc44339639)

[1.6 Approach 4](#_Toc44339640)

[2 Materials & Methods 6](#_Toc44339641)

[2.1 Analysis 6](#_Toc44339642)

[2.2 Prototyping 6](#_Toc44339643)

[2.2.1 Tools Included in the Evaluation 🡪 zu löschen 6](#_Toc44339644)

[2.2.2 Tools Used Outside the Evaluation Part 🡪 zu löschen 6](#_Toc44339645)

[2.2.3 Rational and Set-up 7](#_Toc44339646)

[2.2.4 Used tools 7](#_Toc44339647)

[2.2.4.1 General Set-Up 7](#_Toc44339648)

[2.2.5 Method 9](#_Toc44339649)

[2.3 Audit of the Prototype 9](#_Toc44339650)

[3 Computerised System Validation according to GAMP5 10](#_Toc44339651)

[3.1 GAMP5: An Overview 10](#_Toc44339652)

[3.1.1 GAMP5 and Computerised System Validation 10](#_Toc44339653)

[3.1.2 Key Concepts 10](#_Toc44339654)

[3.1.3 Software Categories 11](#_Toc44339655)

[3.1.4 The Life Cycle Project Phase and its Verification Activities 11](#_Toc44339656)

[3.1.5 Automated Testing 12](#_Toc44339657)

[3.2 Verification for Custom Applications According to GAMP5 12](#_Toc44339658)

[3.3 The OQ Process According to GAMP5 14](#_Toc44339659)

[3.3.1 The Main Process 14](#_Toc44339660)

[3.3.2 Incorporating the Quality Risk Assessment 15](#_Toc44339661)

[3.3.3 Specification- and Test Management 15](#_Toc44339662)

[3.3.4 Exemplary OQ Process 17](#_Toc44339663)

[4 Behaviour Driven Development 19](#_Toc44339664)

[4.1 BDD a suitable Software Engineering Approach for Highly Regulated Environments 19](#_Toc44339665)

[4.2 The Approach: An Overview 19](#_Toc44339666)

[4.3 Writing Executable Specifications with Gherkin 19](#_Toc44339667)

[4.4 Test Automation 19](#_Toc44339668)

[5 OQs using BDD 21](#_Toc44339669)

[5.1 The Combined Process 21](#_Toc44339670)

[5.2 Discussion and Conclusions 21](#_Toc44339671)

[5.2.1 Functional Specification is (partially) fused with the OQ process 22](#_Toc44339672)

[5.2.2 New Elements are Required 22](#_Toc44339673)

[5.2.3 Changes in the Documentation Set-Up 23](#_Toc44339674)

[5.2.4 Final Conclusions Part 1 23](#_Toc44339675)

[5.2.5 Questions araising from part 1 23](#_Toc44339676)

[6 Prototyping 25](#_Toc44339677)

[6.1 System Context and Application Design 25](#_Toc44339678)

[6.2 Architecture of the implemented Prototypical Apps 27](#_Toc44339679)

[6.2.1 The OQ Test App 27](#_Toc44339680)

[6.2.1.1 C4 Model Container Diagram 27](#_Toc44339681)

[6.2.1.2 C4 Model Component Diagram 28](#_Toc44339682)

[6.2.2 The Java Business Application 29](#_Toc44339683)

[6.2.2.1 JBA Frontend 30](#_Toc44339684)

[6.2.2.2 JBA Backend 30](#_Toc44339685)

[6.2.2.3 Database 30](#_Toc44339686)

[6.3 Analysis of the applicability of automation tools in regulated environments in respect of validation. 30](#_Toc44339687)

[6.3.1 Requirements 30](#_Toc44339688)

[6.3.2 Risk-Assessment 31](#_Toc44339689)

[6.3.3 Single Tool Analysis 31](#_Toc44339690)

[6.3.3.1 Scenarioo 31](#_Toc44339691)

[6.3.3.2 OQ Test App 32](#_Toc44339692)

[6.3.4 Analysis of the OQ Test App/Scenarioo System 33](#_Toc44339693)

[6.3.5 Dealing with Updates – Pain Point Glue Code 33](#_Toc44339694)

[6.4 Specification/Formulation 36](#_Toc44339695)

[6.4.1 From User Stories to Feature Files 36](#_Toc44339696)

[6.4.2 Risk Assessment 36](#_Toc44339697)

[6.4.3 Compliance 36](#_Toc44339698)

[6.5 Test Automation 36](#_Toc44339699)

[6.5.1 Glue Code 36](#_Toc44339700)

[6.5.2 Test Reports 37](#_Toc44339701)

[6.6 QA Processes 37](#_Toc44339702)

[6.6.1 Specification and Test Management 37](#_Toc44339703)

[6.6.1.1 Approval 37](#_Toc44339704)

[6.6.2 Traceability 38](#_Toc44339705)

[6.6.3 Test Review 38](#_Toc44339706)

[6.7 Implications for the automated OQ when adding new functionalities 38](#_Toc44339707)

[7 Results of the Prototype Audit 39](#_Toc44339708)

[7.1 xxxx 39](#_Toc44339709)

[7.2 xxxx 39](#_Toc44339710)

[8 Learnings & Discussion 40](#_Toc44339711)

[8.1 xxxx 40](#_Toc44339712)

[8.2 xxxx 40](#_Toc44339713)

[9 Outlook 41](#_Toc44339714)

[9.1 General 41](#_Toc44339715)

[9.2 Specific Topics 41](#_Toc44339716)

[9.2.1 PQs 41](#_Toc44339717)

[9.2.2 IQs 41](#_Toc44339718)

[9.2.3 Category 4 Software 41](#_Toc44339719)

[9.2.4 Agility 42](#_Toc44339720)

[9.2.5 DevOps 42](#_Toc44339721)

[9.2.6 Device Integration 42](#_Toc44339722)

[10 Formatting 44](#_Toc44339723)

[10.1 Quotes 44](#_Toc44339724)

[10.2 Enumerations 44](#_Toc44339725)

[10.3 Footnotes 44](#_Toc44339726)

[10.4 Figures 44](#_Toc44339727)

[10.5 Tables 45](#_Toc44339728)

[10.5.1 Tables as illustration 45](#_Toc44339729)

[10.5.2 Word tables 45](#_Toc44339730)

[10.6 Changing the font 46](#_Toc44339731)

[References 47](#_Toc44339732)

[List of Figures 49](#_Toc44339733)

[List of Tables 50](#_Toc44339734)

[List of Abbreviations 51](#_Toc44339735)

[Appendix 52](#_Toc44339736)

# Introduction

## Initial Situation

To ensure patient safety, software that directly or indirectly influences product quality must be validated according to Good Automated Manufacturing Practices (GAMP) (Wikipedia, 2018; International Society for Pharmaceutical Engineering ISPE, 2008, pp. 14, 15 and 27). This means that a formal and objective proof must be provided that the software is compliant and that its intended use is achieved (International Society for Pharmaceutical Engineering ISPE, 2008, p. 14; Johner, 2017). In support of this validation process, GAMP5 is a guide on how to achieve computerised system validation (CSV) (International Society for Pharmaceutical Engineering ISPE, 2008, p. 14).

According to GAMP5, software validation includes among others, the verification it the user requirements and functional specifications have been met in the software to be introduced (International Society for Pharmaceutical Engineering ISPE, 2008, p. 38).

Until today the wega CSV specialists team experienced that the so called PQs (for verification of the user requirements) and OQs (for verification of the functional specifications) are often performed manually, even though test tools like hp alm (Guru99, 2020a) are supporting testing documentations in regulated companies (Evelyne Daniel, personal communication, December 19, 2019 and April 1, 2020).

From another perspective, Jae Burnett suggests the usage of test automation tools for validated pharmaceutical environments in her paper ‘Practical Use of Automated Tool in Computer System Compliance’: Life science companies should consider automated testing as an opportunity that could add significant value to the computer system compliance process (Burnett, 2009, p. 75).

But she also mentions, that a full test automation might not always be possible, as it might be difficult to integrate the formal approval and the control of test cases into the test automation system (Burnett, 2009, p. 75).

## BDD High Level Test Automation

OQs are considered as high-level testing as they verify that the functional specifications are fulfilled (International Society for Pharmaceutical Engineering ISPE, 2008, p. 38).

Behaviour driven development (BDD), on the other side, is a software development approach that includes tools for high level test automation (Smart, 2015, p.25). One of such a tool is Cucumber (Smart, 2015, p.25). As the test automation script is based on a formalised natural language, that is human and machine readable, it has the potential to be a powerful asset in the sense of a hybrid approach between automation and manual processes (Burnett, 2009, p. 75; Nagy & Rose, 2018, chapter 4.6).

In the same sense, Gáspár Nagy and Seb Rose state in their book ‘Discovery: Explore behaviour using examples’ that BDD is well suited for software development in regulated areas and refer thereby also to the U.S. Food and Drug Administration (FDA) (Nagy & Rose, 2018, chapter 4.6).

* Tools 🡪 einfügen, dass cucumber und Gherkin typische BDD automation tools sind und durch Tools wie Selenium und Scenarioo für die speziell für das automatisierte Testen der UI entwickelt sind, Cucumber/Gherkin which allow to automate the test cases using an automation script (gherkin feature file) (SmartBear Software, 2020).
* Selenium which simulates the user interaction with the web application and will be controlled by cucumber and the gherkin feature file (Selenium, n.d; Jain & Sawant, 2018).
* Scenarioo that is used to display test reports with screenshots (Scenarioo, n.d.).

## Automated Testing for OQ

As described in several blogs, manual testing has the disadvantages that it is prone to mistakes and errors while performing the tests and that it is time consuming for human testers and therefore it generates high costs (Guru99b, 2020; Hoogenraad, 2017). This is especially true, when the same tests have to be performed several times (Guru99b, 2020). On the other hand, and in contrast to automated testing, manual testing allows to evaluate aspects like user friendliness and positive customer experience (Guru99b, 2020). Therefore, this project will focus its investigations on test automation for OQs as testing on functional level does not include evaluations on usability aspects (Qualitest, n.d.).

In conclusion, OQs are interesting to investigate in respect of test automation, as they are so far mainly performed manually and have therefore a good potential of optimisation in respect of reliability and cost effectiveness. And in contrary to PQs there is no need to consider aspects of user friendliness and positive customer experience, for which automated testing is not suitable.

## Hypothesis and Research Questions

This project will be led by following hypothesis: BDD with its activities from user stories to executable specifications (formulation) and automation is a practicable approach in respect of technical feasibility, taking into account Cucumber/Gherkin, Selenium and Scenarioo, and validation requirements according to GAMP5 for OQ test automation in highly regulated environments of the pharmaceutical industry.

To evaluate the above-mentioned hypothesis, the project should find answers to following questions:

* Do the artefacts out of the BDD process satisfy the GAMP5 requirements in respect of OQs?
* Can automation tools like Cucumber/Gherkin, Selenium and Scenarioo (see chapter 2.2.1) be used together in validated environments?
* How can the test suite be adapted to the evolution of the application?
* How could be dealt with new versions of the automation tools in terms of validation?

## Scope

### In Scope

* Evaluation of a test automation for OQs based on BDD for a category 5 software (custom application) according to GAMP5.
* Proposal for a Validation Procedure for Cucumber/Gherkin, Scenarioo and Selenium and their interaction.
* Implementation of a prototype for the evaluation and illustration of a test automation for different functionalities. The test automation is based on the following tools: Cucumber, Selenium, Scenarioo.
* Are to be included: User Requirements, Specification/Test Management, Risk Management, Traceability, the validation process with regard to the OQs for a Category 5 product according to GAMP5.
* If possible, outlooks on related topics (PQs, category 4 software, DevOps), based on the findings obtained.

### Out of Scope

* Validation activities according to GAMP5 outside of OQs (e.g. process validation, IQs, PQs, design reviews, used infrastructure ...).
* BDD activities that are outside the chain from user stories to high-level test automation (e.g. the implementation technique TDD, Unit/Module Testing).
* Tool evaluations for implementation (e.g. Selenium vs. Cypress) are not part of this work.
* Risk evaluation regarding the implementation of the prototype: exemplary risks are considered, but without claiming that the risk evaluation was carried out correctly and completely from a practical point of view.
* Compliance of the prototype: if possible, an exemplary compliance requirement that is technically easy to implement will be considered. However, the prototype will not be compliant to regulations like FDA 21 CFR part 11 or EU GMP Annex 11.

## Approach

The starting point for the BDD part is the book "Discovery - Explore behaviour using examples" by Gaspar Nagy and Seb Rose (ISBN 978-1983591259). In chapter 4.6 the authors state that BDD leads to improved efficiency in software development for regulated environments.

The CSV part of the project is based on the widely used CSV standard GAMP5.

The BDD approach and the GAMP5 methodology are analysed to develop a combined process. This process is then presented in an appropriate form (e.g. BPMN) so that it can be evaluated by a wega CSV specialist. The process will be regularly reviewed by a wega CSV specialist while being developed to assure its conformity with GAMP5. Based on the developed process a prototype will be implemented as proof of concept. Few exemplary user requirements are defined as a basis for its implementation. As the final input regarding the suitability of the developed model, the prototype is audited by a wega CSV specialist. This then forms the basis for the ‘Learnings & Discussion’ and the ‘Outlook’.

In summary following activities are planned in two different streams as shown in the figures 1 & 2:

* Development of a combined process between BDD and GAMP5 in respect to OQs
* Based on point 1 to implementation of a prototype
* Audit of the prototype by wega
* Analysis of the audit results including learnings, a discussion and an outlook
* Analysis of what is needed to be done in order to use Cucumber/Gherkin, Scenarioo and Selenium in a validated environment including following aspects
  + single tools
  + combination of the tools
  + updates of the tools



Figure 1 Process to investigate OQ test automation



Figure 2 Analysis of the applicability of the foreseen automation tools in regulated environments

# Materials & Methods

## Analysis

GAMP5-Analyses

BDD

Analyse über die validierungsmöglichkeiten des Prototypen für die automatische OQ durchführung

Tool Evaluation for Cucumber/Gherkin, Scenarioo and Selenium: An analysis will be performed on what is needed in order to be able to use these tools in a validated environment.

Literature search: Semantic scholar, google, google scholar, swissuniversities databases

BPMN: <http://www.bpmn.org/>

Draw io desktop app from draw.io

## Prototyping

### Tools Included in the Evaluation 🡪 zu löschen

* Cucumber/Gherkin which allow to automate the test cases using an automation script (gherkin feature file) (SmartBear Software, 2020).
* Selenium which simulates the user interaction with the web application and will be controlled by cucumber and the gherkin feature file (Selenium, n.d; Jain & Sawant, 2018).
* Scenarioo that is used to display test reports with screenshots (Scenarioo, n.d.).

### Tools Used Outside the Evaluation Part 🡪 zu löschen

The use of the following tools is anticipated but might be subject to changes. These tools will support the work but are not relevant for the evaluation itself.

* Front end: vue CLI 🡪 <https://cli.vuejs.org/>
* Back end: Spring 🡪 <https://spring.io/>
* Database: PostgreSQL 🡪 <https://www.postgresql.org/>
* Others:  
  Maven 🡪 [https://maven.apache.org/](https://maven.apache.org/guides/introduction/introduction-to-repositories.html)   
  GitHub 🡪 <https://github.com/>

### Rational and Set-up

The prototype will consist on two independent applications. A Business Application and a application to automatically perform OQ over the business application.

### Used tools

#### General Set-Up

The prototype was developed in two Maven (reference) projects: one for the JBA and one for the OQ Test App. Both projects were maintained and are available in a GitHub repository (link). The used libraries and technologies were integrated as described in the pom files of the projects and additionally for the front end in the json file ‘package’(see attachment...). These pom and json files were all kindly provided by Andreas Hosbach and slightly adapted to the present projects. Additional files provided by Andreas Hosbach are the babel.config.js, the postcess.config.js and the vue.config.js file which he provided together with his help to set up the technology stack and the project back-bone, on which these two Apps were build on. The code developed in the course of this project is found in the src folders. Following online resources were used for their development, next to the books referenced in the text:

<https://stackoverflow.com/>

<https://www.guru99.com/>

<https://www.tutorialspoint.com/index.htm>

<https://www.tutorialspoint.com/selenium/index.htm>

<https://cucumber.io/docs/cucumber/>

<https://www.tutorialspoint.com/cucumber/index.htm>

<https://cucumber.io/docs/gherkin/>

<https://www.javatpoint.com/spring-boot-tutorial>

<https://www.javatpoint.com/selenium-tutorial>

<https://www.javatpoint.com/java-tutorial>

<https://vueschool.io/courses?filter=free-courses>

<https://vuejs.org/>

<https://bootstrap-vue.org/>

<https://commonmark.org/help/>

Scenarioo 5.0.2 was deployed as war file with two configuration in respect of the connection to java.exe and the folder with the test results (<http://scenarioo.org/docs/master/tutorial/Scenarioo-Viewer-Web-Application-Setup.html> - 29.6.20)

C4 model for software architecture (<https://c4model.com/>).

Code is based on Code written by Andreas Hosbach (200406 -<https://github.com/andreashosbach/cucumber-reporter> )

OQ TEST APP

* AdoptOpenJDK 14 with HotSpot as JVM (200406 - <https://adoptopenjdk.net/index.html?variant=openjdk14&jvmVariant=hotspot>) was chosen as it is an open-source version of the Java Standard Edition platform (200406 - <https://en.wikipedia.org/wiki/OpenJDK>). There are different Open JDK distributors on the market (200406 - <https://en.wikipedia.org/wiki/OpenJDK>). AdoptOpenJDK was chosen as it is recommended by stackoverflow when no specific environmental or license requirement are needed and the most standard DK build would therefore be appropriate (200406 - <https://stackoverflow.com/questions/52431764/difference-between-openjdk-and-adoptopenjdk>) .
* IntelliJ IDEA 2019.2.4 Community Edition was used as development environment: 200406 - <https://www.jetbrains.com/idea/>
* Maven version 3.6.1 was used to allow dependency management: for description see also the oq-test-app POM on <https://github.com/sableu/BDD4OQ>.......
* JUnit Jupiter is used as testing framework on the Java Virtual Machine (JVM). It is used in the version as described in the oq-test-app POM on <https://github.com/sableu/BDD4OQ>.......
* Cucumber/Gherkin and Selenium (200415 - <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>) were imported via the maven pom file and were used in the versions as described in the oq-test-app POM on <https://github.com/sableu/BDD4OQ>.......
* The Cucumber-Scenarioo-plugin version 0.1.0 was downloaded from (200406 (<https://github.com/andreashosbach/cucumber-reporter>) , installed in the local maven repository by creating a maven build as described (200408 - <https://maven.apache.org/guides/getting-started/index.html>) and integrated as maven dependency as described in the oq-test-app POM on <https://github.com/sableu/BDD4OQ>.......
* Scenarioo release 5.0.2 was setup and used in the standalone application version as described in (200408 - <http://scenarioo.org/docs/master/tutorial/Scenarioo-Viewer-Web-Application-Setup.html#setup-1---running-as-standalone-application>)
* Chrome Browser version 80.0
* Chrome Driver version 80.0 downloaded from (200406 - <https://chromedriver.chromium.org/downloads>) to allow automated testing by Selenium (200406 - <https://chromedriver.chromium.org/>)

### Method

Incremental Software Engineering Approach in order to evaluate the question, what implication the extension of the application with new functionalities on the test suite (see chapter 6.6).

## Audit of the Prototype

The audit will be done by wega. The goal of the audit is to evaluate the documents and the prototype itself that were created within the scope of this project, in respect to the GAMP5 requirements for OQs of a category 5 software. The audit report is attached to this document.

# Computerised System Validation according to GAMP5

## GAMP5: An Overview

### GAMP5 and Computerised System Validation

GAMP5 is a worldwide used industry guidance on computerised system validation in the pharmaceutical industry ([https://ispe.org/pharmaceutical-engineering/may-june-2018/gamp-5-ten-years 200621](https://ispe.org/pharmaceutical-engineering/may-june-2018/gamp-5-ten-years%20200621) Wochenende vom 20.6.). It has been developed by an international group of experts from the International Society of Pharmaceutical Engineering, ISPE ([https://ispe.org/pharmaceutical-engineering/may-june-2018/gamp-5-ten-years 200621](https://ispe.org/pharmaceutical-engineering/may-june-2018/gamp-5-ten-years%20200621) Wochenende vom 20.6.).

The goal of the guideline is to provide assistance in achieving Good Automation Practice (GAMP) by ensuring that computerised systems are fit for their intended use and compliant (GAMP5, p11). The process to achieve this goal and to provide the corresponding proof in a documented form is called ‘Computerised System Validation’, CSV (Johner Institut).

### Key Concepts

GAMP5 is based on 5 key concepts (GAMP5, p19) that will be described in the following:

* Understanding of the product and the process: To ensure fitness for intended use it is fundamental to understand the product and the process to allow the correct definition of the requirements for the system (p.19).
* Consideration of the whole life cycle: In order to guarantee the fitness for intended use and to assure that compliance is maintained at any time, the whole life cycle of the system has to be taken into account. GAMP5 distinguishes between the concept-, the project, the operation- and the retirement phase (GAMP5, p.19 in Verbindung mit p26).
* Scalable activities over the whole life cycle: Depending on factors like the impact, the novelty or the complexity of the system, csv activities should be scaled accordingly (p.20).
* Science Based Quality Risk Management: The quality risk management is ensured through a systematic process to determine the critical aspects of the computerised system. The risks need to be managed, controlled and reduced to an acceptable level (p.20).
* Leveraging Supplier Involvement: Suppliers of computerised systems have knowledge, experience and documentation about their products. The purchaser (called regulated company in GAMP5) should make use of it to reduce his csv efforts to a minimal level (p.21).

### Software Categories

As seen in the chapter before, risk management and scalable life cycle activities are important concepts in GAMP5. Both concepts are reflected in the GAMP5 categorisation of software products (GAMP5, Appendix M4, p127. – p 131).

GAMP5 distinguishes between following four categories, omitting category 2(p. 128 – p. 130) :

* Infrastructure (Category 1)
* Non-Configured Software (Category 3)
* Configured Software (Category 4)
* Custom Software (Category 5)

From Category 1 to 5 the risk usually increases due to higher complexity and less user experience (p.127), thereby increasing the required csv activities (GAMP p.130 und 131).

### The Life Cycle Project Phase and its Verification Activities

The most interesting life cycle phase to investigate in respect of verification activities and test automation using BDD is the project phase as BDD is a software development approach (Discovery p. 55) and because this phase consist on four stages with one of them entirely dedicated to verification as shown in the following (p.29):

* Planning
* Specification, Configuration, and Coding
* Verification
* Reporting and Release

These four stages are foreseen for all software types, but the extend of each step varies depending on their category (p.29 - 37). For example the verification step of a category 3 software consists only of requirements testing (p.34), whereas for a category 5 software it includes module testing, integration testing, functional testing (corresponding to the OQ, as described in chapter 3.3 ) and requirements testing.

Since we want to investigate the automation of OQ, we will focus in the following on a category 5 software.

Next to verification activities in the project phase, verification activities are also needed in the operation phase while implementing changes in the software (GAMP5, figure 4.1 page 30). But GAMP5 states in this respect that as well the verification stage as also the other three stages of the project phase are equally applicable for the subsequent changes during operation (p.29).

### Automated Testing

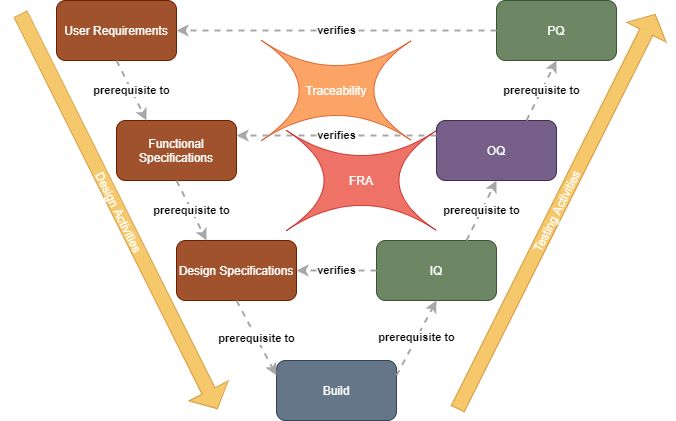
GAMP5 has a chapter specifically dedicated to automated testing (GAMP5 pp207). It is stated, that automated testing offers a good possibility to improve efficiency and effectiveness of test execution especially in respect of test coverage for regression testing (p207). But it also states some rules that should be respected when using test automation tools. With regard to the research questions of this thesis, the following points should be emphasized:

* The tools need to be validated (appropriate specification and risk based verification), whereas GAMP5 normally considers established test tools as a category 1 software (p207)
* The use of test automation tools should be defined in the test strategy and utilised in accordance with its intended use (207).
* The maintenance of test documentation have to conform at least to the standard for paper based testing (207, 208): To control automated test scripts, a documented procedure must be created and applied (208). The test results are normally available as logs generated by test automation. These should not only containt the acutal test results, but must also contain information such as an ID, date and time of test execution, the connection to the corresponding test script, the identity of the tester and the name of the test environment. They should furthermore be available for reviews and audits in a state that prevents the documents from being editable. The QA needs to agree on the handling of the test result documentation. This should be done while developing the test strategy. (208)

## Verification for Custom Applications According to GAMP5

Verification activities for the implementation of a custom application are about demonstrating that the software is compliant and fit for intended use by confirming that specifications have been fulfilled (GAMP5 pp. 31 and 37). GAMP5 foresees, that after the application has been built, an installation qualification (IQ), an operational qualification (OQ) and a performance qualification (PQ) is achieved by testing activities (GAMP5 p 38). In this sense a test strategy, also called testplan, should define and document among others, how IQ, OQ and PQ should be applied for the specific software, based on company procedures that were established to define the general framework for testing (GAMP5 pp. 196 and 201). On p.196, GAMP5 foresees, that the testplan is written by the test manager, whereas for the approval the Quality Unit is suggested (p.59).

The figure below shows in an abstract form how specifications are related to the different testing activities (GAMP5, p. 36; 200417 - <https://www.softwaretestinghelp.com/iq-oq-pq-software-validation/> ; 200419 - <https://www.blazesystems.com/blaze-iq-oq-pq.html> ; 200419 - <https://www.ciprecision.com/validation-services/>; 200427 – auch heruntergeladen - <https://onlinelibrary.wiley.com/doi/pdf/10.1002/qaj.426> ):



In view of OQ automation, it has to be noted, that the OQ is performed on the fully built and installed software. In order to respect this process, that OQ is performed after installation, it will not be possible to use automation tools that perform testings during the build process, as for example unit testing using Junit (look for reference). Therefore, an OQ Test App will be needed, that uses the interfaces of the deployed application.

It must also be pointed out that traceability between the individual specifications (from user requirements to functional specifications and design specifications) as well as to the corresponding tests must be ensured throughout the entire process (GAMP5, pp. 134-137).

In a similar way than traceability, also functional risk assessment (FRA) is based or has an impact on the the whole software verification process (GAMP5 Page 51).

## The OQ Process According to GAMP5

GAMP5 defines OQs the following way: “Operational Qualification (OQ)” [...is a...] “documented verification that a system operates according to written and pre-approved specifications throughout specified operating ranges (GAMP5 p. 38). Whereas GAMP5 foresees following verification activity: “Testing or other verification of the system against specifications to demonstrate correct operation of functionality that support the specific business process throughout all specifies operating ranges” (GAMP5 p. 38). In the GAMP5 appendix D5 p. 212 for custom applications, this is explicitly linked to functional testing, as it states that these testings should focus on functionality that supports the specific business process based on risk and supplier assessment which exactly corresponds to the wording used to describe OQs as seen before.

### The Main Process

Writing OQs may start in parallel with the development of the functional specifications by the supplier (e.g. the IT unit of the company (🡪 check for reference), by elaborating the corresponding test specifications (GAMP5 p199 in combination with p175). GAMP5 does not mention which role is foreseen to write the test specifications. The test specifications describe the overall purpose and a description of a set of test scripts (GAMP5 p198). For example, it defines which resources are needed, including tools for automated testing, the version of software under test, the test scripts to be carried out, methods, prerequisites, required reviews and approvals, etc. (GAMP5 p199). In addition, GAMP5 requires some metadata about the test specification document (GAMP5 p199).

Based on the test specification, the test analyst is responsible for developing test scripts that describe the tests to be performed in such a way that the testers can execute them consistently (GAMP5 p199). Next to some metadata like unique test reference and cross references to control specifications, test scripts consist also of a title, a test description, the test objective, prerequisites, test steps, acceptance criteria and instructions about data to be recorded (GAMP5 p199 f.).

GAMP5 foresees, that the test specifications and the test scripts can be recorded in a single document (GAMP5 p 200). From this statement it could be deduced, that the test analyst is not only responsible for the test script elaboration, but could also be assigned by the test manager to write the test specifications, if it is not the test manager him- or herself who will do it (GAMP5 p 196).

As already mentioned, the test scripts are executed by the testers (GAMP5 p 196). As a result of their testing, they have to deliver records that are reviewable. The records contain the result of the single tests (passed/failed) including needed descriptions and supporting documentations as defined in the test scripts for example screen shots (GAMP5 p 200).

Subsequent to the execution of the tests, the test results will be reviewed by the test reviewer, which should not be the same person as the tester. Based on this review a test report will be delivered by the test reviewer (GAMP5 p. 196 and p. 200). A test report includes again some metadata like who executed and who reviewed the testings, and information about the effected testings like a summary of the test results, a summary of test failures and conclusions (GAMP5 p 200).

A Figure that shows which documents are produced by whom during this process

### Incorporating the Quality Risk Assessment

Quality risk management is one of the five key concepts in GAMP5 (GAMP5 p.20). It is an iterative process that covers the entire life cycle of a computerised system (p.47 GAMP5). In this sense, it has also an important role to play in the above described OQ process for which it is considered to be a supporting process (GAMP5 p32). The goal of this concept is to focus validation efforts on critical points of the computerised system (GAMP5 p.20). The quality risk management process includes the identification of functions with impact on patient safety, product quality and data integrity based on an initial risk assessment to determine system impact (GAMP5 p.107). To do this, lies in the responsibility of a team consisting of subject matter experts and key users (p106), or if regulatory compliance is concerned, it is the quality unit (p106). For a next step, this team performs a functional risk assessment and identify controls, based on the advice of the supplier, to eliminate or at least mitigate the risk to an acceptable level (p48 in combination with 50 and 106). Appropriate controls, i.e. quality critical requirements (p. 164), need then to be implemented GAMP p.50, e.g. by the supplier for additional software functionalities (p19) and verified (GAMP p.50) e.g. in the OQ process (p.38).

As the goal of OQ is the documented verification of software functionalities (GAMP p.38 in combination with p212), it is important to take the identified risks, their risk level and the defined control measures in respect to software functionalities into account (GAMP p.50). The specific level of test efforts should than be determined according to the risk level and the system impact (GAMP p.50) and the controls themselves might be subject to the OQs (GAMP5, p38). The risk assessment documentation needs finally be approved by the system owner and/or the quality unit (p106)

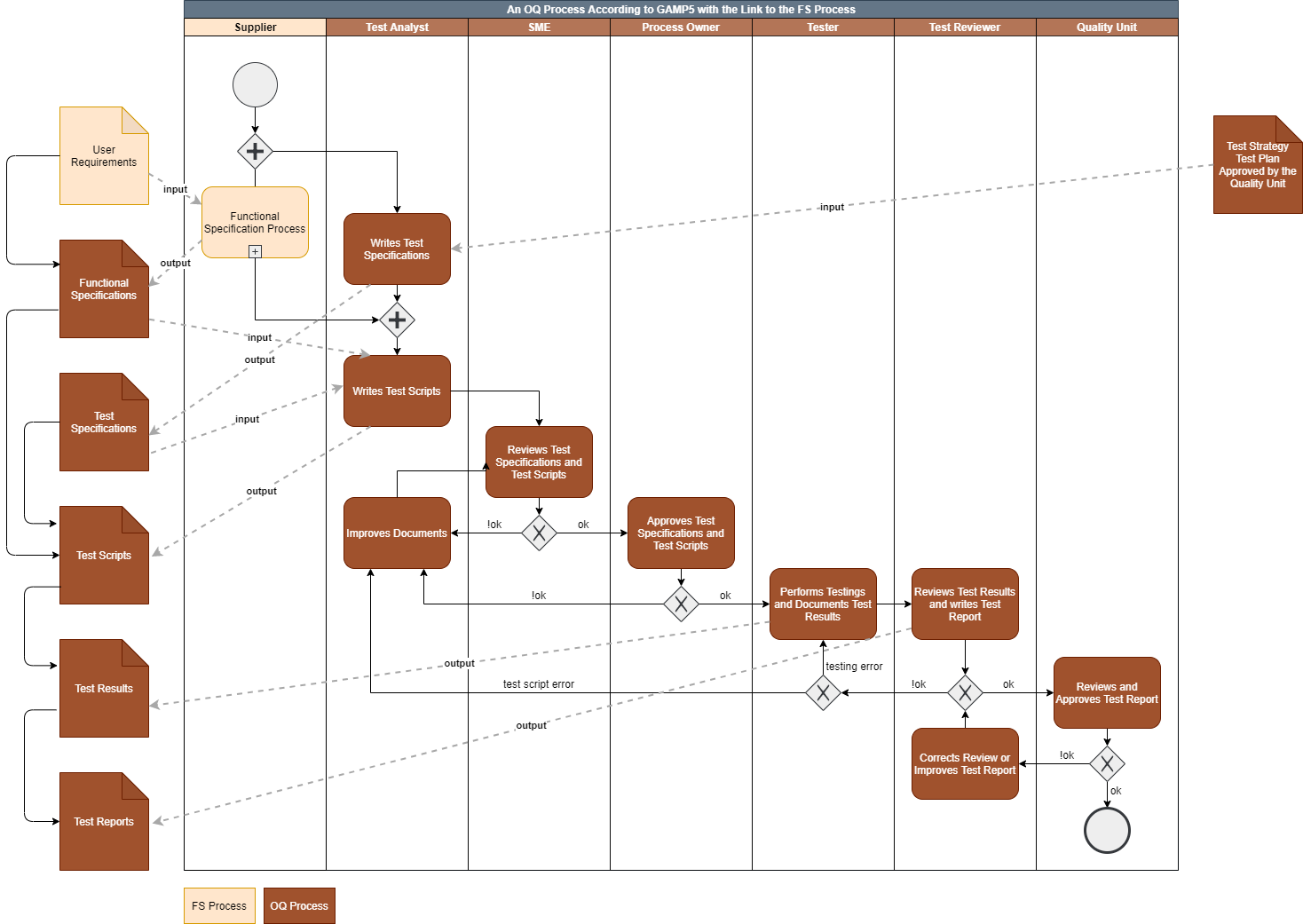
### Specification- and Test Management

Next to the quality risk assessment, there are other supporting processes to be considered in respect to OQs for a custom application (p32). They include:

* Change management process[[1]](#footnote-1): “Change management procedures also should be established. The point at which change management is introduced should be defined. Appropriate change processes should be applied to both project and operational phases.” (p.32). While performing OQ no change of the software is expected, as it is done on the version for which in a previous step the IQ were performed and approved (p.209). Therefore, the change management process will only be considered in the sense, that the tester has to state on which version the OQ is performed.   
  An OQ Result might be, that the software under test did not pass this qualification. Normally, that will result in a new version of the software, as a fix will have to be introduced. This new version will again be submitted to an IQ and an OQ, during which the approved tests will be re-run by documenting the new software version (fig chapter 3.2.)..
* Configuration management: “Appropriate configuration management processes should be established such that a computerized system and all its constituent components can be identified and defined at any point” (p.32). Bringing this back to the level of the OQ process, it has to be clearly stated on which version of the software the OQ is performed, i.e. the same version as the preceding IQs (p.209).
* Traceability is the process to ensure that requirements are covered and traced to the corresponding functional specifications and design components, which then must be linked further to the appropriate verification (p.33; p.134ff). This means for the OQ process, that each OQ test script needs to be traced back to the underlying functional specification, which on its turn needs to have a link to the requirements from which it is derived.
* Document management process: “Management of documentation includes preparation, review, approval, issue, change, withdrawal, and storage”. (p. 33). The process described by GAMP5 (p. 153ff.) can be adapted in order to fit to the complexity of the project (p. 153). In respect of the OQ process following points were identified to be the most important ones:
* The author is normally responsible for the document prior to its review and the document, which normally should be subject to version control, is in the status ‘draft’ (p154).
* The draft is then reviewed ideally by an independent Subject Matter Expert (SME) for the specific field and the subsequent actions should be resolved prior to approval and issue (p.154; p60).
* The approval of a documents consists of a signature, a comment about the approval reason and a date (p. 154). The document index and - history should be updated and the new status set, i.e. from ‘draft’ to ‘approved’ (p.154). According to the role description of GAMP5, this could be the Process Owner in respect to the test specifications and the test scripts, as she/he is responsible for the system (fitness for intended use and compliance) and the quality unit in respect to the test report (p.58; p. 196).
* The approved document is being issued by updating the document index (p. 154). This could mean in respect to the OQ process, that the approved test scripts are handed over to the tester.
* Document changes needs to by controlled by, e.g., updating the document index and -history and by setting the status back to ‘draft’, thereby and in consequence repeating the document approval process as described before (p. 155). In respect to the OQ, it might be a result of a test execution, that there are test script errors and therefore the concerned test scripts will be reset to ‘draft’ and corrected before being again approved.
* Document withdrawal can be done by updating the document index, -history and status and information of any controlled copy holders (p.155)
* In respect of document records and storage GAMP5 states that they should be stored in a safe and secure way according to a defined process (p.155).

### Exemplary OQ Process

Based on the descriptions on the earlier chapters, an exemplary process was developed, which highlights the most important tasks, roles and documents in respect to the OQ process. In view of the BDD process, the link to the Functional Specification Process is also taken into account.



# Behaviour Driven Development

Literature:

* The BDD Books: Discovery – Explore behaviour using examples   
  Gáspár Nagy and Seb Rose   
  ISBN: 978-1983591259
* BDD in Action – Behavior-Driven Development for the whole software lifecycle  
  John Ferguson Smart  
  ISBN: 9781617291654
* Writing Great Specifications – Using Specification by Example and Gherkin  
  Kamil Nicieja  
  ISBN: 9781617294105

## BDD a suitable Software Engineering Approach for Highly Regulated Environments

## The Approach: An Overview

## Writing Executable Specifications with Gherkin

* Basierend auf dem Buch: Writing Great Specifications: Using specification by example and Gherkin von Kamil Nicieja

## Test Automation

BDD Automation Tool:   
- Cucumber: <https://cucumber.io/>

Further tools to complement Cucumber:  
- Scenarioo: <http://scenarioo.org/>  
- Selenium: <https://www.selenium.dev/>

# OQs using BDD

## The Combined Process

It was tried to define an exemplary process independent of the software development process (agile, waterfall, etc.). It could be all defined at the beginning of the project like in a waterfall or one could go through this process in several itarations, coming closer to an agile approach.

Formulation Team: Should consist on persons that do well know the practicle side of the Requirements (User, Regulatory and Quality Critical requirement, a tester, but not the persons having the tester role, as they should be independent, and one or several representatioves of the IT Unit).

## Discussion and Conclusions

In principle it is possible to define an OQ process according to GAMP5. It does not need to be exactly the way as was shown before, but with this process we have a poof of concept, that using BDD could be done in a way that is compatible with GAMP5

Viel vernetzter, prozess der functional specifications entfällt, traceability sehr gut garantiert, aufwertung des Testers der auch eine Qualitätssicherungsfunktion übernimmt, die manuelle ausführung des tests (stupide Arbeit entfällt)

Ob Prozes weniger Aufwand bringt ist fraglich, aber stupide Arbeit entfällt und wird statt dessen hoffentlich in bessere Kontrolle und Qualität investiert 🡪 zum Beispiel überienstimmng der Feature Files und der Glue Files 🡪 Quality check.

Es lohnt sich ein prototype zu bauen um ein besseres Gefühl zu erhalten, auch wenn der Prototyp nur bedingt nach diesem PRozess entwickelt werden kann, da es kein Team hat.

The BDD Elements of interests in respect of test automation could be neatly integrated in the OQ Process

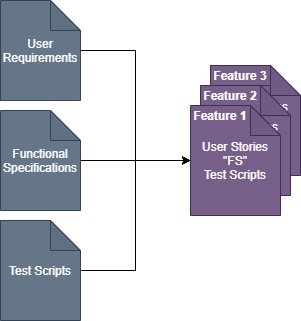
### Functional Specification is (partially) fused with the OQ process

* Describe the most important elements of the FS-Process to show that fusion is not entgegen einer GAMP5 Anforderungen  
  🡪 Testable ist gut  
  🡪 Rollen gehen auf  
  🡪 Traceability ist gewährleistet
* Zu Überdenken: FS Dokument beinhaltet auch Schnittstellen Beschreibungen und Design Constraints 🡪 es gilt zu überlegen ob der eigentliche Teil der Functional Specification vom den beiden anderen Punkten getrennt werden soll
* Functional testing aber auch testing der Facilities 🡪 könnten das auch Geräte sein (z.B. automated store?) 🡪 hier ist es out of Scope, aber im Outlook abhandeln, wie mit Geräteintegration umgegangen werden könnte.

### New Elements are Required

* Tester Role is different
* Implication on Test Specifications

### Changes in the Documentation Set-Up



Good news for Traceability

### Final Conclusions Part 1

Es konnte nichts gefunden wrden, was dem Einsatz von BDD automation tools entgegensprechen würden. Im Gegenteil, einige Punkte lassen vermuten, dass es einfacher wird über die automation hinaus

* Traceability
* Less redundancy durch zusammenlegung vo FS und OQ Prozess
* Aufwertung tester mit zusätzlichem Qualitätscheck.

### Questions araising from part 1

* Zu überprüfen: wenn software nach BDD entwickelt wird, basierend auf den Tests, was hat das für einen Einfluss auf den OQ Prozess?
* Document approval und versionierung 🡪 wie geht man damit um

# Prototyping

Based on the findings as described in the previous chapters, a prototype was created to provide a proof of concept, or to highlight the GAMP5 requirements for which OQ automation might fail.

Explicitly excluded from prototyping is the creation of SOPs and a test plan. Where necessary for the understanding, aspects normally found in SOPs or in the test plan was introduced in the test specification or are found in this thesis.

Further excluded was the validation of the test automation tools. Nevertheless, some thoughts about the validation of the tools will be described as foreseen according to the research question in the introduction (🡪 link auf Kapitel).

The prototype is based on a fictive situation with fictive players (even if real names were used) even though a scenario was used, that could be part of a clinical trial scenario especially in respect to the business requirements.

The business app (JBA) was only implemented to the point to show how a specific GAMP5 requirement could be fulfilled. There is no functionality that would be ready to be used in a real situation.

Coding guideline : Parallel versionierung

Elektronisches Dokuemntenmanagement

Signaturen out of scope

Ziel des Audits ist sicher zu stellen, das die OQ einer App automatisch durchgeführt werden.

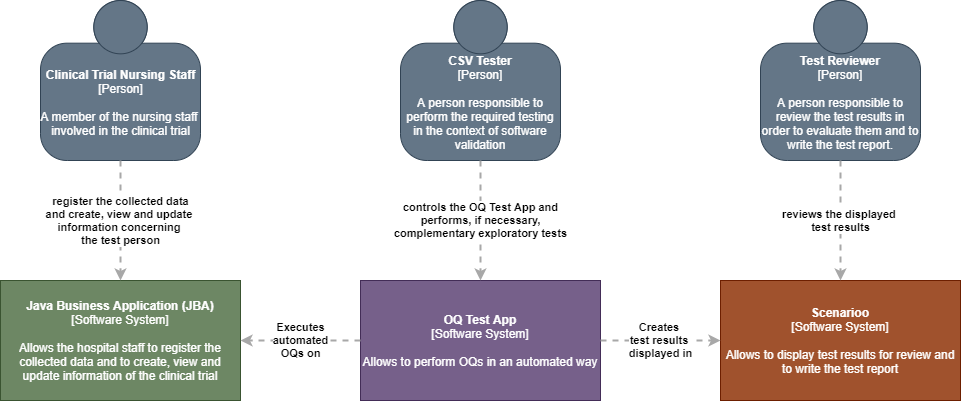
Aufzunehmen: Kompare Funktion in der Diskussion 🡪 What turned out during the audit was the compare function, which is an interesting tool for the QA (review by exception)

## System Context and Application Design

The prototype consists of three independent applications as shown in Fig. .

These three applications could be described as follows:

* The Java Business Application (JBA) was implemented as representation of web-based business application to support the management of a fictive clinical trial. From the business point of view, this is the app of interest that needs to be validated.
* The OQ Test App was implemented as a prototype to perform automated OQs on the JBA based on the feature files and the glue code. It generates the testresults in a format, that allows to be displayed in Scenarioo.
* Scenarioo is an open source software available and documented on <http://scenarioo.org/>. The software was designed for automated documentation of user interface tests. As it is not specifically foreseen for the regulated environment of the pharmaceutical industry, it is not validated yet.



System context of the Prototype Design according to the C4 model

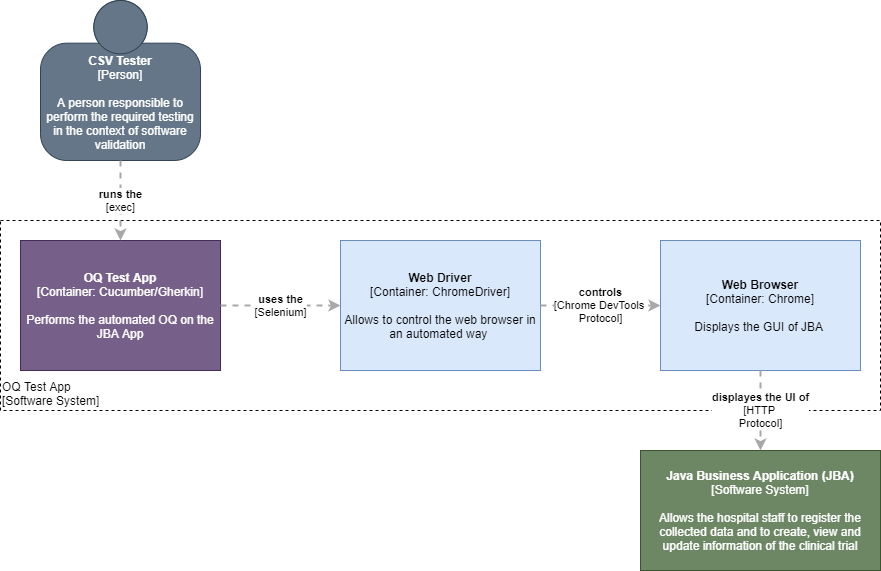
## Architecture of the implemented Prototypical Apps

### The OQ Test App

The OQ Test App was designed to perform the OQ of the JBA. It is based on the JUnit Jupiter test framework with the cucumber test runner to perform the automated test. The architectural design was developed based on the C4 model (<https://c4model.com/> - 27.6.20) and is shown in the two following chapters.

#### C4 Model Container Diagram

When the tester runs the OQ Test App, the JBA will be tested in an automated way by making use of the web driver ChromeDriver, which therefore needs to be installed.

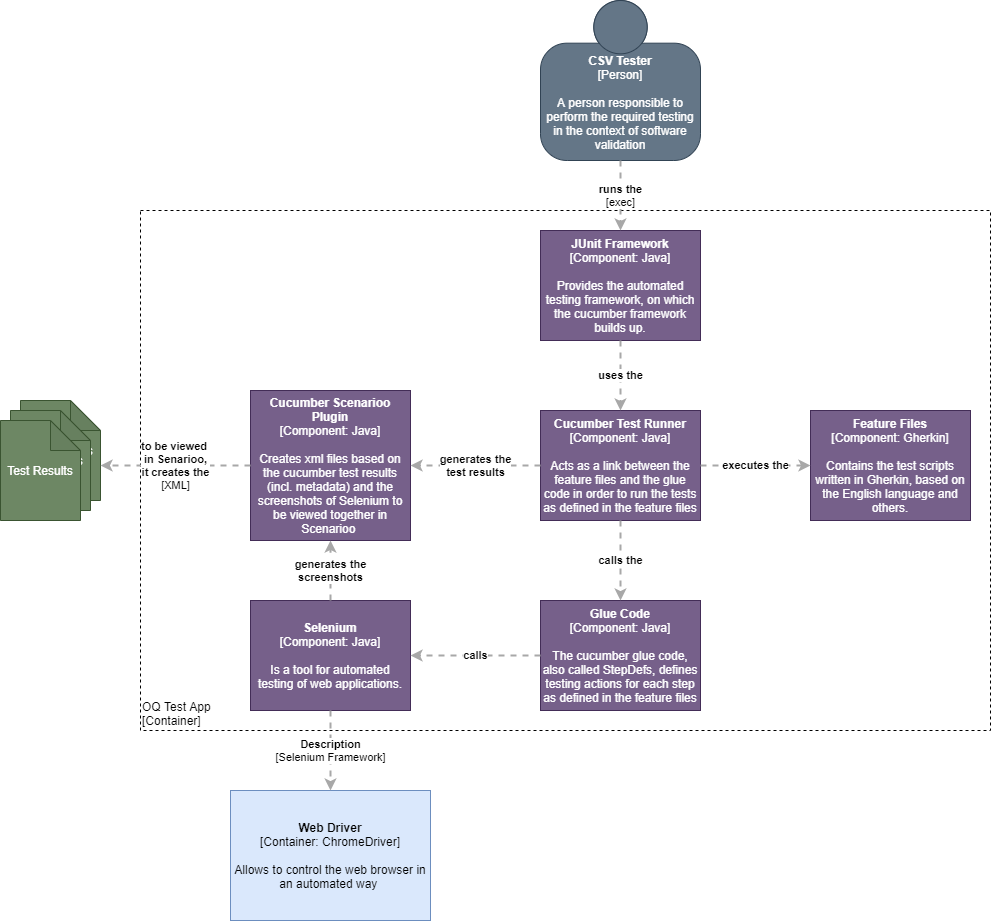


In order to make it work, the web driver corresponding to the respective chrome version needs to be installed and referenced to in the OQ Test App. For the prototype chrome version 83 and chromeDriver version 83.0 were used (<https://chromedriver.chromium.org/downloads>)

Reference for communication between chromedriver and chrome: <https://groups.google.com/forum/#!msg/chromedriver-users/xVMy5OGLcl8/2JljtZ1FAAAJ> viewed the 27.6.2020

#### C4 Model Component Diagram

When we zoom into the OQ Test App Container, as shown in the figure above, we can get of view of the components on which the OQ Test App is build. The components are shown and described in following figure:



The description in the diagram was based on following references (viewed the 28.6.2020):

<http://www.automationtestinghub.com/cucumber-test-runner-class-junit/>

<https://www.tutorialspoint.com/cucumber/cucumber_junit_runner.htm>

<https://www.coveros.com/exploring-glue-code-with-cucumber-jvm/>

<https://www.tutorialspoint.com/cucumber/cucumber_gherkins.htm>

<https://www.tutorialspoint.com/selenium/selenium_webdriver.htm>

<https://github.com/andreashosbach/cucumber-reporter>

### The Java Business Application

The Java Business App (JBA) was designed as web-application on which the automated OQ should be performed. It represents a custom application for a fictive clinical trial in which the weight of the registered participant needs to be measures before starting the test treatment (baseline weight measurement). It includes one functionality to demonstrate how to deal with risks (registration of the baseline weight measurement) and one functionality to demonstrate how to deal with functionalities needed in order to be compliant (consent). Additionally, all implemented functionalities serve to test the overall GAMP5 OQ requirements for automated testing.

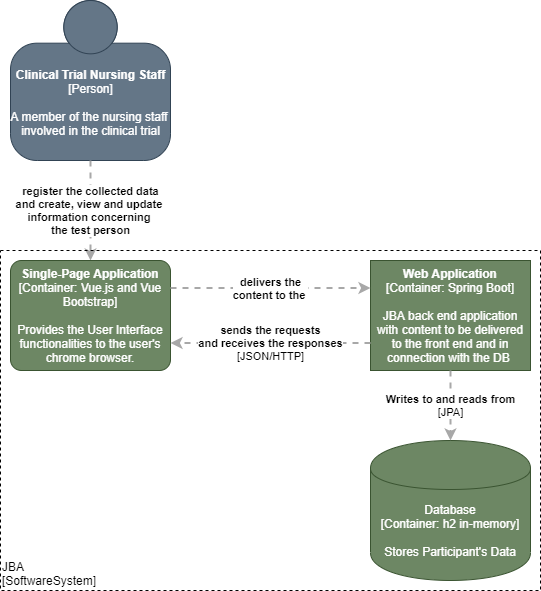


Fig: Container Diagram according to the C4 model of JBA

#### JBA Frontend

The JBA Frontend was designed as a vue.js single page application (<https://vuejs.org/> - 28.6.20). In addition the user interface styling was based on the BootstrapVue components (<https://bootstrap-vue.org/docs/components> - 28.6.20).

#### JBA Backend

For building the backend, spring boot was used as it is designed to create stand-alone Spring based applications that provided in an embedded way the Apache Tomcat HTTP web server environment ( <https://spring.io/projects/spring-boot> - 28.6.20).

#### Database

As a database, the H2 Java SQL in-memory database was used (<https://spring.io/projects/spring-boot> - 28.6.20) . It is a relational database management system that can be embedded in Java applications (<https://en.wikipedia.org/wiki/H2_(DBMS)> - 28.6.20).

## Analysis of the applicability of automation tools in regulated environments in respect of validation.

In order to analyse the research questions as defined in the introduction, following approach was defined in chapter 16:”

* Analysis of what is needed to be done in order to use Cucumber/Gherkin, Scenarioo and Selenium in a validated environment including following aspects
  + single tools
  + combination of the tools
  + updates of the tools”

This will be done in order to respond to the research questions as defined in chapter....(links to introduce) . Based on the defined architecture in the chapter 6.2, this analysis can now be done.

### Requirements

As for any tool in the regulated environment of the pharmaceutical industry, the OQ Test App with its components needs to be validated (compare chapter 1.1 first paragraph) (🡪 Link einfügen). The OQ Test App is part of the infrastructure of the test environment for the business applications submitted to an OQ. In consequence this means that the productive environment of the OQ Test App is the test environment of the JBA and before being productively used in that environment, verification activities needs to be performed dependent on the category and taking into account the risk, the complexity and the novelty of the software (GAMP5 p.33-p.37; p.32).

### Risk-Assessment

Validation activities are based on the risk due to the usage of the tools in respect of their intended use. The intended use of the final OQ Test App/Scenarioo System is the automated OQ testing of the business application.

Two adverse effects of a mal-functioning OQ Test App/Scenarioo System could be identified in respect of its intended use, meaning a fully tested JBA in respect of its functionalities:

* The first effect that could come into play, would be if the OQ Test App would not perform all required functional tests as defined in the test script and that this would not be detected. The result out of that, would be, that the JBA could be deployed to production with an uncomplete OQ. As there is a real probability that this could happen if the OQ Test App/Scenarioo System is not working properly, there is indeed a risk that needs to be mitigated.
* The second situation that should be avoided is, that tests that should fail would not be detected as such and that in consequence they would be considered to have passed. This would mean that the JBA could be deployed to production with functionalities that would not fulfil the functional specification. In this situation there is a real probability that this could happen if there is a respective bug in the OQ Test App/Scenarioo System. Therefore, this risk needs to be mitigated to an acceptable level also in respect of this adverse effect.

There were more adverse effects detected, but these adverse effects would not result in a risk. Such an adverse effect was for example, that a test would fail instead of pass. No risk due to this effect in respect of fitness for intended use could be deduced as the consequence would be a recheck of the concerned functionality of the JBA where the outcome would be, that the functionality was fine, but the OQ Test App had a bug.

### Single Tool Analysis

#### Scenarioo

In order to validate Scenarioo, following aspects have to be taken into account:

The war deployed Scenarioo 5.0.2 could be considered as non configured product even though minimal installation configuration was required to indicate the path to the folder with the test results from the OQ Test App (<http://scenarioo.org/docs/master/tutorial/Scenarioo-Viewer-Web-Application-Setup.html> - 29.06.20). Most importantly, there were no configuration needed in respect of functionalities or other interfaces next to the test result folder. Therefore, it would be regarded in principle as a category 3 software according to GAMP5 (p. 128-129)

As foreseen for a category 3 software (p. 108), a functional risk assessment based on the over all risk assessment in chapter 3.6.2 (p50) would be needed to be performed for validation of the software.

And finally in respect of leveraging supplier involvement (p.19), no indication was found on the scenarioo web page, that scenrioo would have been used so far in a controlled environment of the pharmaceutical industry (<http://scenarioo.org/> – 29. Even though leverage of supplier involvement might still be of some value, one has to assume, that the main validation activities would need to be done by the regulated company.

For the moment nothing could be detected, that would impede the validation of Scenarioo and therefore its usage in a GxP environment.

#### OQ Test App

The OQ Test App is a custom application and therefore a category 5 software according to GAMP5 requirering the highest level of validation efforts (p.129 and 130). What might leverage somewhat these efforts is the fact, that at the component level, the software is mainly based on publicly available libraries (GAMP5 p 127) and that it has in fact very little own content, that consists basically on the feature files and the glue code.

As was shown in the chapter 6.2.1.2 the OQ Test App consists on following components:

* The JUnit Framework
* The Cucumber Test Runner
* Gherkin as part of Cucumber to write the feature files
* The Glue Code
* Selenium
* And the Cucumber Scenarioo Plugin

The Junit Framework, the Cucumber Test Runner, Gherkin and Selenium are established testing tools (<https://www.stickyminds.com/article/junit-vs-testng-choosing-framework-unit-testing#:~:text=JUnit%20is%20one%20of%20the,Selenium%20WebDriver%20tests%20in%20Java.> <https://cucumber.io/tools/cucumber-open/> <https://en.wikipedia.org/wiki/Cucumber_(software)> <https://www.browserstack.com/guide/selenium-webdriver-tutorial> – all viewed the 29.6.20 ) and could therefore be considered as category 1 software[[2]](#footnote-2) according to GAMP5 (p. 207).

As the Glue Code and the Cucumber Scenarioo Plugin were specifically developed and adapted, respectively, they need to be considered as category 5 software (p.129 and 130).

With this in mind and taking into account the risks as described in the chapter 6.3.2, nothing could be determined, that would make it impossible to validate the OQ Test App and therefore its usage in a GxP environment. But quite some efforts will be required for its validation, as it has to be considered as category 5 software as described before.

### Analysis of the OQ Test App/Scenarioo System

As already described earlier (chapter....), the OQ Test App generates test results and formats them using the Cucumber Scenarioo Plugin in order that the test results can be visualised in Scenarioo. The most critical point in respect of the OQ Test App/Scenarioo integration is therefore the correct functioning of the Cucumber Scenarioo Plugin of the OQ Test App. With this in mind, the integration of the OQ Test App/Scenarioo System should be part of the validation of the OQ Test App as described in the chapter before.

### Dealing with Updates – Pain Point Glue Code

In principle, any change in the software needs a re-validation of the new software version (GAMP5, Figure 4.1 on page 30). For both software, the OQ Test App and Scenarioo, this can be fully controlled: It depends on the user needs, when changes in the applications are needed, as both applications are on premise and in full control of the regulated company.

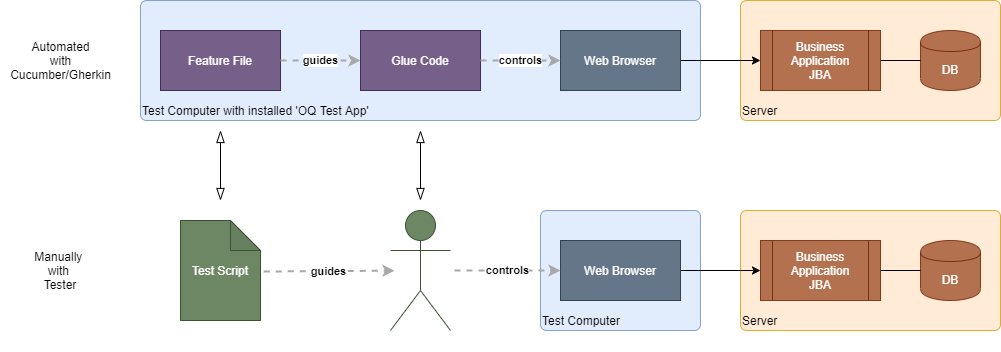
For scenarioo this therefore means that the new version can be validated and installed, when new features would add so much new value to the users that it is worth the required validation efforts of the new Scenarioo version and possibly also the required changes in the cucumber-scenarioo plugin of the OQ Test App.

The situation of the OQ Test App is a bit different compare to the situation for Scenarioo: For each new OQ of the JBA, changes in the feature files, and more importantly in the glue code, become necessary. This means in its consequence that for each OQ the OQ Test App would need to be re-validated as the Glue Code is part of the OQ Test App source code. Taking into account, that the OQ Test App is a GAMP5 category 5 software, this would lead to considerable validation efforts for which it is not sure, that they could be justified by the expected gain of reduced effort due to the OQ automation.

A possible solution to that could be to ‘outsource’ the glue code from the rest of the OQ Test App and to concentrate the validation of the OQ Test App on the remaining component, as they are the stable core of the application. This has three consequences: The remaining OQ Test App core consists on category 1 components, which are considered to be reliable and the cucumber-scenarioo plugin on which the validation efforts would need to be concentrated on. The second consequence is that the OQ Test App would need an additional functionality consisting of importing the feature file and the glue code, thereby compiling and integrating the glue code into the OQ Test App. It can be assumed, that there is no real risk coming from this new functionality, as in case of a bug, the OQ Test App would rather run into an error state than giving wrong test results. The third consequence is, that the correctness of the glue code needs to be verified in a different way than it would be done for software validation as it is not a software but just a file with code.

For the further course of this project, it will be assumed in respect of the prototype, that the above described validation concept of the OQ Test App core would be followed, even though if for the prototype the glue code was directly build into the OQ Test App. But the glue code verification process was developed, as if the glue code would need to be imported into the validated and deployed OQ Test App core.

The rationale that was developed in order to allow the glue code verification is summarised in figure ..... :



The function and the intended use of the glue code is to control the web-client (i.e. the web-browser) of the JBA based on the feature files. Figuratively speaking, the glue code assumes the function of the tester in a manually performed OQ testing.

In order to do so, the glue code uses methods provided by Selenium which in turns uses the web-driver corresponding to the foreseen web browser. This makes it possible to control the JBA web-client like a tester would do but in an automated way, as for example:

* Fields can be filled in with test data provided by the feature file.
* Buttons can be clicked
* Contents of a web page can be analysed
* Screenshots can be taken

The tester is error prone as it has also to be assumed for the not verified glue code. Therefore, the tester needs to take screenshots and write some descriptions as testing evidence so that during the subsequent test review, the performed tests can be verified. This same method could also be used to verify the correct functioning of the glue code. Therefore, Scenarioo has an important role to display a test description, some data for evidence on the test performance and screen shots. If one could fully rely on the glue code the standard built-in cucumber test report could be sufficient.

This implies, that the screen shots have to be taken in a way, that the correct functioning of the glue code can be checked. This can be achieved by checking following points:

* No empty functions in the glue code. This can be checked in two ways:
* In the code review of the tester
* By the test reviewer who has to verify, that each step took at least few milliseconds to perform.
* The clue code only performs actions on the UI and this in the same way a human tester would do. Selenium is a tool to test UIs in an automated way. These tests are only meaningful, if Selenium performs only actions a user would do. Therefore, Selenium only supports UI actions users could perform in the same way (this could be verified in an OQ of Selenium  not done so far). By only using Selenium to perform the UI actions, the human tester can be simulated. The tester has to make sure in the code review, that only Selenium functions are used by the glue code in this sense. The tester has to make sure, that the glue code does not make use of any other interface, unless if explicitly specified. For a web application, there is the risk of making directly a rest call. But this would be easy to identify.

## Specification/Formulation

### From User Stories to Feature Files

### Risk Assessment

Example Mapping 🡪 red risk card was added to the one defined theoretically to include this in the three amigos meeting.

### Compliance

Consent Management as an example 🡪 Process owner is responsible 🡪 will chose SME that have the know-how 🡪 he es the owne who defines the SME 🡪 see GAMP5 his role description. They will bring it in during the requirements and in the three amigos meeting they will assure, that the right examples will be found and assigned in order that compliance ist umgesetzt.

## Test Automation

### Glue Code

Glue code only uses Selenium for testing. ....

Remark: In the Glue Code of the prototype the rest api is called at one point. It is used to clean the database from the data of the preceding step and is not directly part of the testing itself. It has the advantage that it makes eacht scenario completely independent from the other, but it might be an artefact for easier prototyping and in a real project it might be, that it should be defiend in a differen way 🡪 coding guidelines, handling of test data.

### Test Reports

## QA Processes

### Specification and Test Management

#### Approval

OQ Documents needs to be approved, this is especially true for feature files. How it turned out, feature files will after injecting them into the implementation process change. Therefore a solution has to be found how it could be done. Feature files are a mixture of the functional specification and the test script. According to GAMP5 the functional specifications needs to be approved before injecting them into the implementation process 🡪 but this is based on GAMP5 According to Evelyne Daniel, no legal regulation, that has to be followed strictly, whereas GAMP5 is ‘just’ a guide and considered as good practice but which says itself, that it can be changed as far as it is still compliant to regulation. For the testing part, the documents have to be approved, before OQ starts, as this is after implementation, this can be done on the finelised documents either manually or electronically. This should be defined in the Test Plan.

If after creation of the feature files, changes needs to be tracked despite of the above mentioned findings, a easy way could be using Git with its Version control and a Jira Plugin, to allow the business to access the relevant versions in Git, without having to master Git, whisch seems not to be obvious for the business. The Review and Approval by the business could then be managed in Jira.

### Traceability

### Test Review

## Implications for the automated OQ when adding new functionalities

# Results of the Prototype Audit

## xxxx

## xxxx

# Learnings & Discussion

## xxxx

## xxxx

Coding Guideline fpr JBA: Unterstützen durch BDD Prozess auch in der Entwicklungsphase, Design der Weboberfläche die sowohl benutzerfreundlich ist, wie auch, z.B. durch kontraste, die den Revew Prozess durch den Test Reviewer / QA vereinfachen

Compare Funktion:

# Outlook

## General

Selenium Testing Vor- Nachteile: Aufwändig wenn sich Oberfläche ändert, OQs können einfach auf verschiedenen Browsern ausgeführt werden 🡪 Testen der Applikation auf verschiedenen Browsern.

Digitization: Prozess möglichst klassisch gehalten, so wie er jetzt, manuell schon ist. Test automation passt sich diesem Prozess an. Denkbar wäre, OQs schon im Build-Prozess (inside Testing vs. Outside testing) durchzuführen. Würde aber bedeuten, dass die Reihenfolge IQ 🡪 OQ nicht mehr stimmen würde und dass die Qualificationen der Auditoren und der QA Menschen sich verschieben müssten 🡪 Müssten Code lesen und beurteilen können (vgl. Kleines BDD Buch)

## Specific Topics

### PQs

### IQs

### Category 4 Software

### Agility

E.v. mit DevOps verbinden

### DevOps

### Device Integration

# Formatting

## Quotes

Enable „Replace "Straight quotes" with “smart quotes”” in the AutoFormat section of the AutoCorrect options. Default language is „English (U.K.)“.

Variant: «…»

## Enumerations

Please use the „List Bullet“ template.

* Item 1
* Item 2

## Footnotes

Example for text containing a footnote.[[3]](#footnote-3)

## Figures

Please use the following option to insert graphical illustrations (e.g. Power Point charts):

Home Tab 🡪 Paste 🡪 Paste Special… 🡪 Picture (Enhanced Metafile).

Use the template „Figure“ for the formatting of illustrations. After inserting the illustration it must be anchored to the text. For this purpose, place the cursor in the target area of the figure (a paragraph which is formatted as „Figure“), do a right click on the figure and follow the instructions below:

Size and position 🡪 Text Wrapping 🡪 In Line with Text

The size of the illustration can also be changed in „Size and position“.

To **name** your illustration, place the cursor below your figure and choose „Insert Caption“ from tab „References“.

Caption: Figure number, colon, space, caption.

Please compare the following example: Figure 1



Figure 3: Example illustration

## Tables

### Tables as illustration

For tables which are imported as **illustration** or **object** e.g. from excel, use the same procedure like for figures (2.4) except that the illustration is formatted as “Figure Table” instead of “Figure”

For the caption use the same label “Table” as for word tables and format it as “Caption Table”.

Table caption: Below the table, like in our example: Table 1



Table 1: Example for a table imported as illustration

### Word tables

Choose our template „Table heading“ for title row and „Table text“ for table content. Please follow our instructions to caption your table:

Tab “References” 🡪 “Insert caption” (Position: Below selected item)

Reformat accordingly to „Caption Table“ (Styles).

Caption: Table number, colon, space, caption.

|  |  |
| --- | --- |
| Bezeichnung der  Grössenklasse | Anzahl Beschäftigte  (Vollzeitäquivalent) |
| Kleinunternehmen | 10 bis unter 50 |
| Mittlere Unternehmen 1 | 50 bis unter 100 |
| Mittlere Unternehmen 2 | 100 bis unter 250 |

Table 2: Example for a table created by Word

## Changing the font

The font for the whole document can be changed by the “normal” template (is not recommended). All other styles which are used in this document are linked to this template.

To change the font anyway:

Tab „Home“ 🡪 „Styles“ 🡪 Choose template „Normal“ in the dropdown-menu (right click) 🡪 „Modify“

References

Burnett, J. (2009). Practical Use of Automated Tools in Computer System Compliance. Journal of Validation Technology, 15(4), 73–76. Retrieved from https://www.ivtnetwork.com/journal-validation-technology/journal-of-validation-technology-3004

GitHub, Inc. (2020). Build for developers. Retrieved March 27, 2020, from https://github.com/

Guru99. (2020a, March 19). Introduction to HP ALM(Quality Center). Retrieved April 3, 2020, from https://www.guru99.com/hp-alm-introduction.html

Guru99. (2020b, March 23). Automation Testing Vs. Manual Testing: What’s the Difference? Retrieved April 3, 2020, from https://www.guru99.com/difference-automated-vs-manual-testing.html

Hoogenraad, W. (2017, October 11). Was ist manuelles Testen? Retrieved April 3, 2020, from https://de.itpedia.nl/2017/10/11/wat-is-handmatig-testen/

International Society for Pharmaceutical Engineering ISPE. (2008). GAMP 5 - A Risk-Based Approach to Compliant GxP Computerized Systems. North Bethesda, USA: ISPE.

Jain, N., & Sawant, T. (2018, February 5). Setup for Selenium with Cucumber Using Maven. Retrieved April 3, 2020, from https://www.axelerant.com/resources/team-blog/setup-for-selenium-with-cucumber-using-maven

Johner, C. (2017, April 5). Computer System Validation CSV. Retrieved March 29, 2020, from https://www.johner-institut.de/blog/regulatory-affairs/computer-system-validation-csv/

Nagy, G., & Rose, S. (2018). Discovery - Explore behaviour using examples. Victoria, Canada: https://leanpub.com/.

Nicieja, K. (2018). Writing Great Specifications. Shelter Island, NY, USA: Manning Publications.

Object Management Group. (1997 to present). Business Process Model and Notation. Retrieved March 27, 2020, from http://www.bpmn.org/

Qualitest. (n.d.). Functional Testing vs. Usability Testing. Retrieved April 3, 2020, from https://www.qualitestgroup.com/white-papers/functional-testing-vs-usability-testing/

scenarioo. (n.d.). Retrieved March 27, 2020, from http://scenarioo.org/

Selenium. (n.d.). Selenium automates browsers. That’s it! Retrieved March 27, 2020, from https://www.selenium.dev/

Smart, J. F. (2015). BDD in Action. Shelter Island, NY, USA: Manning Publications.

SmartBear Software. (2020). Tools & techniques that elevate teams to greatness. Retrieved March 27, 2020, from https://cucumber.io/

The Apache Software Foundation. (2002 to present). Maven. Retrieved March 27, 2020, from https://maven.apache.org/

The PostgreSQL Global Development Group. (1996 to present). PostgreSQL: The world’s most advanced open source database. Retrieved March 27, 2020, from https://www.postgresql.org/

VMware. (2020). Spring makes Java... Retrieved March 27, 2020, from https://spring.io/

Wikipedia. (2018, June 26). GxP. Retrieved March 29, 2020, from https://de.wikipedia.org/w/index.php?title=GxP&oldid=178646512

You, E. (2018 to present). Vue CLI. Retrieved March 27, 2020, from https://cli.vuejs.org/

List of Figures

[Figure 1 Process to investigate OQ test automation 4](#_Toc36826148)

[Figure 2 Analysis of the usability of automation tools for regulated environments 5](#_Toc36826149)

[Figure 3: Example illustration 19](#_Toc36826150)

Annotation: A convenient way to create default list of figures and list of tables in the text is to follow the sequence:

Tab „References“ 🡪 „Captions“ 🡪 „Insert Table of Figures“

Now, the caption is formatted automatically according to our template „caption“.

For the figures the template „Figure“ and for tables the templates „Table heading“ and „Table text“ should be used.

Further information can be found in chapters: 2.4 and 2.5

List of Tables

[Table 1: Example for a table imported as illustration 3](#_Toc416160904)

[Table 2: Example for a table created by Word 3](#_Toc416160905)

List of Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Description |
| BDD | Behaviour Driven Development: A software development approach developed by Dan North covering the whole software lifecycle |
| CSV |  |
| FDA | U.S. Food and Drug Administration ….. |
| GAMP |  |
| GAMP5 |  |
|  |  |
| GxP |  |
| JBA |  |
| JVM | Java Virtual Machine |
| OQ | Operational Qualification: …. |
| PQ | Performance Qualification …. |
| SME | Subject Matter Expert |
|  |  |
|  |  |
|  |  |

Appendix

References and appendix are formatted as „Heading 1 without numbering.“ Thereby this entry is listed in the table of contents (but without numbering). To structure your appendices use CAPITAL LETTERS (Appendix A, Appendix B etc.).

1. Personal Communication Evelyne Daniel the 25.4.2010 : Normally the change management process starts with the IQ [↑](#footnote-ref-1)
2. Personal Communication by Evelyne Daniel the 26th of June 2020: It has to be defined on a case by case base, but she would tend to assume a category 3 for established testing tools and not a category 1 as stated by GAMP5 [↑](#footnote-ref-2)
3. Example Footnote. [↑](#footnote-ref-3)