University of Applied Sciences Northwestern Switzerland
School of Business

Student Project

BDD - A Practicable Approach for Computerised System Validation?

Bachelor Thesis 2020

Client: wega Informatik AG
Author: Sabrina Leuenberger
Lecturer: Stephan Jüngling
City, Date: Murten, 23rd 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 has not been published to date. It has thus not been made available to other interested parties or examination boards.

City, Date

CH-3280 Murten, 23rd of July 2020

Signature

Acknowledgments

I would like to thank wega Informatik AG, represented by Daniel Juchli, for their commitment to host this project and to support me in its realisation. In particular, I would like to thank Evelyne Daniel and Mathias Fuchs from wega, who took the time for professional discussions and clarifications as well as for input to the project management. They supported me in weekly meetings and proofread the documents to provide important input.

Furthermore, Evelyne Daniel conducted the audit on the prototype and wrote the associated audit report. This report, with its evaluation of the prototype by an experienced CSV specialist, gives the present work a value that goes beyond that of a mere student project - many, many thanks!

Another important and valuable contribution to the success of this project was made by Andreas Hosbach, business analyst at Zühlke Engineering AG. I would therefore like to thank him most heartfully. He helped me to set up the technology stack of the prototype. In particular, he provided several configuration files and code examples on which I could base the implementation of the prototype. He was always willing to help me when I encountered technical problems. The use of Scenarioo and the H2 database was also based on his recommendation, which has proven to be very rewarding.

Furthermore, I would also like to thank the FHNW Olten for the good and well-founded business informatics education and for the approval of this project. A special thanks goes at this point to Stephan Jüngling, who has supported me since the first idea for this project and has worked towards its approval. Many thanks also for the intermediate discussions and all the given inputs during the project.

Moreover, I would like to thank the following persons:

- Andreas Wicki, CSV specialist at wega, for the proofreading and his inputs to the audit report
- Damian Schraube, CSV specialist at wega, for the exchange of ideas and thoughts regarding Cucumber automation in the CSV area
- Manuel Kohler, Software Engineer at wega, for providing the Confluence Space and the Jira project
- Sandro Ibig, Software Testing specialist at ergon, for his input on test strategies using automation tools to complement manual testing and the helpful links to Selenium tutorials
- Stéphane Bisinger, from the University of Bologna, for his suggestions to use Vue.js as a java script front-end framework.
- And last but not least, Delia Bianchi, a business informatics fellow student, for her friendship and for the many hours spent together over the last four years.

Management Summary / Abstract

Computer systems in the highly regulated GxP environment of the pharmaceutical industry must be validated[1] (Computerised System Validation – CSV). This means that formal proof must be provided that these systems are fit for their intended use and compliant to the applicable regulations.

One aspect of validation is the verification process, which includes the verification of the functional specifications (operational qualification - OQ). Even today, this process is still mostly done manually and is therefore time-consuming and error-prone.

Therefore, the aim of this thesis was to examine whether automation of OQ could be implemented in a compliant way and especially in conformity with GAMP5, a widely used guideline in the industry. For this purpose, the Behaviour Driven Development (BDD) approach was used, as it employs practices and tools that enable automated high-level testing.

Three steps were taken to achieve this goal:

1. Development of an OQ process containing the BDD practices to perform automated OQ.
2. Implementation of this process by means of a prototype to check its feasibility.
3. Audit of the prototype by an experienced CSV expert to assess the GxP suitability of the process.

As it turned out, it was indeed possible to develop a process with automated OQs and implement a GxP-compliant prototype based on it. This opens up new possibilities for software validation, which could also be interesting in terms of a digital transformation strategy. Therefore, it seems to make sense to continue on this path and, if possible, to carry out further and more detailed clarifications directly with a wega customer through a real software validation project. This would provide more insights regarding the potential and possible added value (increased efficiency, higher quality, shorter time-to-market) of such automation.

[1] FDA Definition: Establishing documented evidence which provides a high degree of assurance that a specific process will consistently produce a product meeting its pre-determined specifications and quality attributes (US FDA 1987)

Table of Contents

1 Introduction

1.1 Initial Situation

In the pharmaceutical industry, software that directly or indirectly affects the quality of their products must be validated to ensure patient safety. (ISPE, 2008, pp. 14, 15 and 27). This means that a formal and objective proof must be provided that the software is compliant to applicable regulations and that its intended use is achieved (ISPE, 2008, p. 14; Johner, 2017). In support of this validation process, GAMP5 is a guide on how to achieve computerised system validation (CSV) issued by the International Society for Pharmaceutical Engineering (ISPE, 2008, p. 14).

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

Until today the wega CSV specialist's 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 Burnett 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).

1.2 BDD High Level Test Automation

OQs and PQs are considered as high-level testing as they verify that the user requirements and the functional specifications are fulfilled (ISPE, 2008, p. 38).

Behaviour driven development (BDD), in turn, is a software development approach that includes tools for high level test automation (Smart, 2015, p.25). As additionally, the test automation script is based on a formalised natural language, that is easily human and machine readable, it further 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, pp. 72-74).

Moreover, Gáspár Nagy and Seb Rose noted that BDD is well suited for software development in regulated areas and also refer to the US Food and Drug Administration FDA (Nagy & Rose, 2018, pp. 72-74).

Consequently, BDD seems to be a promising approach to achieve high-level test automation for OQs or PQs.

There are different BDD high level automation tools available (Ketterlin Fisher, 2019). But for the purpose of this project, the focus will be laid on following tools:

- Cucumber und Gherkin which are typical BDD automation tools. They allow to automate the test cases using an automation script that is also understandable by business stakeholders without any particular technical skills (SmartBear Software, 2020).
- Selenium which simulates the user interaction with the Web application and is controlled by Cucumber and Gherkin (Selenium, n.d; Jain & Sawant, 2018).
- Scenarioo and the Scenarioo-Cucumber-plugin that were used to display test reports with screenshots that allow a combined process with an automated and a manual part (Scenarioo, n.d.-a; Hosbach, 2020).

1.3 Automated Testing for OQ

As just seen, there are different types of software testing, such as OQs and PQs. In this respect, it must also be taken into account that test automation is not necessarily equally suitable for every type of test.

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 therefore generating high costs (Guru99b, 2020; Hoogenraad, 2017). This is especially true, when the same tests have to be performed several times e.g. in regression testing (Guru99b, 2020). In such situations test automation can add considerable value (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).

As OQs, unlike PQs, do not include evaluations on usability aspects (Qualitest, n.d.; ISPE 2008, p.38), OQs are more promising to investigate for test automation. Additionally, formal testing will often be based on the functional specifications which are often considered to be a contractual document (ISPE, 2008, p. 175).

1.4 Hypothesis and Research Questions

This project was based on 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, this project was in particular investigating following questions:

- Do the artefacts from the BDD-OQ process satisfy the GAMP5 OQ requirements?
- How can the executable requirement suite (see chapter 4.4.3) be adapted to the evolving nature of the application?
- Can automation tools like Cucumber/Gherkin, Selenium and Scenarioo be used together in validated environments?
- And how could be dealt with new versions of the automation tools in terms of validation?

1.5 Scope

GAMP5 covers the whole life cycle of a software for different software categories (ISPE, 2008, p.25, pp. 33-37). It was therefore not possible to consider all aspects of GAMP5 within the scope of this project, which meant that the actual scope had to be clearly defined.

1.5.1 In Scope

Following aspects were addressed within the scope of this project:

- Evaluation of a test automation for OQs based on BDD for a category 5 software (custom application) according to GAMP5.
- Evaluation on the feasibility of validation for Cucumber/Gherkin, Selenium, Scenarioo, Cucumber-Scenarioo-plugin and their interactions.
- Implementation of a prototype for the evaluation and illustration of a test automation. The test automation is based on the following tools: Cucumber, Selenium, Scenarioo and the Cucumber-Scenarioo-plugin.
- User Requirements, specification/test management, risk management, traceability, the validation process with regard to the OQs for a Category 5 product according to GAMP5.
- An outlook and recommendations based on the findings obtained.

1.5.2 Out of Scope

Following aspects were explicitly excluded from scope, as they would go beyond the extend of this project:

- Validation activities according to GAMP5 outside of OQs (e.g. process validation, IQs, PQs, design reviews and 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 (e.g. Selenium vs. Cypress).
- 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: An exemplary compliance requirement is considered. However, the prototype will not be compliant to regulations like FDA 21 CFR part 11 or EU GMP Annex 11.

1.6 Approach

The starting point for the analysis of the BDD approach was the book "Discovery - Explore behaviour using examples" by Nagy & Rose (2018). In chapter 4.6 the authors state that BDD leads to improved efficiency in software development for regulated environments (Nagy & Rose, 2018, pp. 72–74).

The analysis of the requirements in respect of CSV was fully based on the widely used CSV standard GAMP5 (ISPE, 2008).

The BDD approach and the GAMP5 methodology were analysed to develop a combined process. This process was then visualised in a ‘Business Process Model and Notation’ (BPMN) diagram in order to be evaluated by a wega CSV expert (Object Management Group, 1997). The combined process was reviewed by a wega CSV expert while being developed to assure its conformity with GAMP5. Based on the developed process a prototype was implemented as proof of concept. Few exemplary user requirements were defined as a basis for its implementation. As the final input regarding the suitability of the developed model, the prototype was audited by a wega CSV expert. The audit results were analysed and discussed. Based on the gained insights, an outlook and a recommendation were presented to wega Informatik AG.

In summary following activities were performed as shown in the Figure 1 and Figure 2:

1. Development of a combined OQ process between BDD and GAMP5, reviewed by Evelyne Daniel, a wega CSV expert.
2. Based on point 1 implementation of a prototype.
3. Audit of the prototype by Evelyne Daniel.
4. Analysis of the audit results including learnings, a discussion and an outlook
5. Evaluation on the feasibility to validate Cucumber/Gherkin, Scenarioo and Selenium in order to be used in a GxP environment, thereby focusing on:
    - 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 intended automation tools in regulated environments

2 Materials & Methods

2.1 Analysis

A large part of the work consisted of making analyses. In particular, the analysis of GAMP5 with its requirements and an analysis of the BDD practices should be mentioned here, but also the analysis of the audit report prepared within the scope of this project by wega.

To visualise the results of the analysis, BPMN was used (Object Management Group, 1997) and adapted where necessary to provide a specific focus, especially colour highlighting and dependencies in relation to documents. The illustration was done in draw.io (draw.io, n.d.).

2.2 Collaboration and Project Management

This project was carried out in close cooperation with wega Informatik AG, especially with Evelyne Daniel and Mathias Fuchs. Among other activities, weekly meetings took place to manage the project in accordance with wega. To support this cooperation the following tools were used, which were also provided by wega:

- Microsoft Teams (Microsoft, n.d.)
- Atlassian Confluence (Atlassian, n.d.-a)
- Atlassian Jira (Atlassian, n.d.-b)

2.3 Prototyping

2.3.1 Rational

The prototype was built from three independent applications:

- A Java Business App (JBA): This is the Web application on which the OQ was performed
- The OQ Test App: It performed the OQ of the JBA on an automated basis.
- Scenarioo, an existing open source software to visualise test results from automated GUI testing (Scenarioo, n.d.-a).

2.3.2 Used tools

Several tools were used with different purpose as described in the following:

Modelling:

To model the JBA and the OQ Test App, the C4 model for software architecture was used (Brown, n.d.). The models were illustrated with the help of draw.io (draw.io, n.d.).

Implementation:

The following tools were used to implement JBA and the OQ Test App:

- IntelliJ IDEA 2019.2.4 Community Edition was used as development environment (Jet Brains, 2019).
- AdoptOpenJDK 14 with HotSpot as JVM was chosen as it is an open-source version of the Java Standard Edition platform (AdoptOpenJDK, n.d.; Wikipedia, n.d.). There are different Open JDK distributors on the market (Wikipedia, n.d.). AdoptOpenJDK was chosen as it is recommended by stackoverflow when no specific environmental or license requirement are needed and the most standard JDK build would therefore be appropriate (stackoverflow, 2018).
- The whole project was managed in the following GitHub Repository (GitHub, Inc., 2020): https://github.com/sableu/BDD4OQ

Web-Browser

JBA was developed as a Web application using Chrome Browser version 83.0. To enable automation, Chrome Driver version 83.0 was also installed on premise (Chromium, n.d.).

Scenarioo

Scenarioo 5.0.2.war was deployed as described by the provided Scenarioo documentation (Scenarioo, n.d.-b). One configuration has been made to set the path to the folder with the test results (Scenarioo, n.d.-b). It was used in the standalone application version as described in the Scenarioo Version 5.0 documentation (Scenarioo, n.d.-b).

2.3.3 Development of JBA and the OQ Test App

The two applications that needed to be developed, were implemented as maven projects (The Apache Software Foundation, 2002). Both projects were maintained and are available in the GitHub repository found on the following site https://github.com/sableu/BDD4OQ. The used libraries and technologies were integrated as described in the Project Object Model (POM) files and additionally for the front end in the json file ‘package’ (see Appendices II & III). These pom- and json files were all kindly provided by Andreas Hosbach and slightly adapted to the present projects. Additional files from 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 built-on.

The code developed in the course of this project can be found in the src folders in the GitHub repository https://github.com/sableu/BDD4OQ.

Several internet sites turned out to be very useful, when technical support was needed for implementation. These sites were collected and can be found in the Appendix I. Additionally, 'The Cucumber for Java Book' was used (Rose, Wynne, & Hellesøy, 2015).

2.3.4 Implementation Approach

The JBA and the OQ Test App were implemented incrementally. This allowed on the one hand to answer the research question in respect of extending the application with new functionalities (see chapter 6.8) and on the other hand it enabled a step-by-step improvement of the OQ Test App in particular, based on inputs from wega, to achieve GxP compliance.

2.4 Audit of the Prototype

The audit was conducted by wega. The goal was to evaluate the process documents and the prototype itself with regard to the GAMP5 requirements for OQs of a category 5 software. The audit report is attached in Appendix XIV, where more information can be found.

3 Computerised System Validation according to GAMP5

3.1 GAMP5: An Overview

3.1.1 GAMP5 and Computerised System Validation

GAMP5 is a worldwide used industry guidance on computerised system validation in the pharmaceutical industry (Wyn, 2018, p. 18). It has been developed by an international group of experts from the International Society of Pharmaceutical Engineering, ISPE (Wyn, 2018, p. 18).

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 to applicable regulations (ISPE, 2008, p. 11; Wikipedia, 2018). The process to achieve this goal and to provide the corresponding proof in a documented form is called 'Computerised System Validation', CSV (Johner, 2017).

3.1.2 Key Concepts

GAMP5 is based on 5 key concepts (ISPE, 2008, p. 19) 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 in order to allow the correct definition of the requirements for the system (ISPE, 2008, 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 (ISPE, 2008, p. 19, p.26).
- **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 (ISPE, 2008, 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 (ISPE, 2008, 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 (ISPE, 2008, p. 21).

3.1.3 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 (ISPE, 2008, pp. 127- 131).

GAMP5 distinguishes between following four categories, omitting category 2 (ISPE, 2008, pp. 128-130):

- 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 (ISPE, 2008, p. 127), thereby increasing the required CSV activities (ISPE, 2008, pp. 130-131).

3.1.4 The Life Cycle Project Phase and its Verification Activities

To investigate verification activities and test automation using BDD, the most interesting life cycle phase is the project phase, as BDD is a software development approach (Nagy & Rose, 2018, p. 55). This phase consists of four stages with one of them entirely dedicated to verification as shown in the following (ISPE, 2008, p. 29):

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

These four stages are intended for all software types, but the extend of each step varies depending on their category (ISPE, 2008, pp. 29 - 37). For example the verification step of a category 3 software consists only of requirements testing (ISPE, 2008, 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 the automation of OQ is investigated, the focus will be on a category 5 software.

In addition to the verification activities in the project phase, verification activities are also required in the operation phase when implementing necessary changes to the software (ISPE, 2008, p. 30, Figure 4.1). GAMP5 states that the verification activities of the project phase are equally applicable with regard to implementation activities within the operation phase (ISPE, 2008, p.29).

Thus, the findings on the automated OQ process can be applied both to the project phase and to implementation activities in the operation phase, even though the focus of these investigations are on the project phase.

3.1.5 Automated Testing

GAMP5 has a chapter specifically dedicated to automated testing (ISPE, 2008, pp. 207-208). 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 (ISPE, 2008, p. 207). 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 emphasised:

- The tools need to be validated (appropriate specification and risk-based verification), whereas GAMP5 normally considers established test tools as a category 1 software (ISPE, 2008, p. 207).
- The use of test automation tools should be defined in the test strategy and utilised in accordance with their intended use (ISPE, 2008, p. 207).
- The maintenance of test documentation has to conform at least to the standard for paper-based testing (ISPE, 2008, pp. 207-208): To control automated test scripts, a documented procedure must be created and applied (ISPE, 2008, p. 208). The test results are normally available as logs generated by the test automation (ISPE, 2008, p. 208). These should not only contain the actual 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 (ISPE, 2008, p. 208). They should furthermore be available for reviews and audits in a state that prevents the documents from being editable. The Quality Assurance (QA) needs to agree on the handling of the test result documentation. This should be done while developing the test strategy (ISPE, 2008, p. 208).

3.2 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 (ISPE, 2008, p. 31, p. 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 (ISPE, 2008, p. 38). With this in mind, a test strategy, also called test plan, 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

(ISPE, 2008, p. 196, p. 201). On page 196, GAMP5 foresees, that the test plan is written by the test manager, whereas for the approval the Quality Unit is suggested (ISPE, 2008, p. 196, p.59).

Figure 3 below shows in an abstract form how specifications are related to the different testing activities (Plagiannos, 2015; ISPE, 2008, p. 36; Esch et al., 2007, p. 213; Blaze Systems, n.d.):

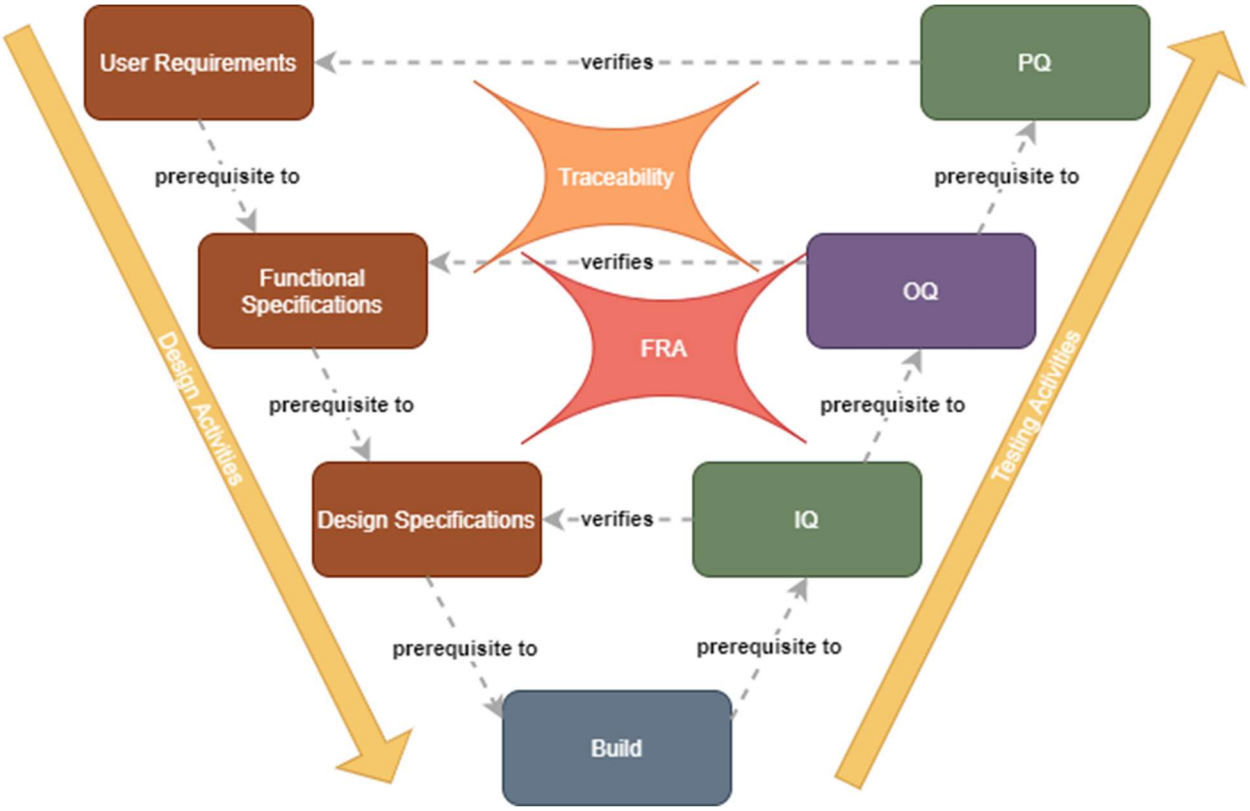


Figure 3: Design- and verification process according to GAMP5

In view of OQ automation, it has to be noted, that the OQ is performed on the fully built and installed software.

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 (ISPE, 2008, pp. 134-137).

In a similar way than traceability, also functional risk assessment (FRA) is based or has an impact on the whole software verification process (ISPE, 2008, p. 51).

3.3 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” (ISPE, 2008, p. 38). Whereas GAMP5 foresees for verification activities: “Testing or other verification of the system against specifications to demonstrate correct operation of functionality that support the specific business process throughout all specified operating ranges” (ISPE, 2008, p. 38). In the GAMP5 appendix D5 for custom applications, this is explicitly linked to functional testing, as it states that these testing 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 (ISPE, 2008, p. 212).

3.3.1 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), by elaborating the corresponding test specifications (ISPE, 2008, p. 175, p. 199). GAMP5 does not mention which role is intended to write the test specifications. The test specifications describe the overall purpose and a description of a set of test scripts (ISPE, 2008, p. 198). 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. (ISPE, 2008, p. 199). In addition, GAMP5 requires some metadata about the test specification document (ISPE, 2008, p. 199).

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 (ISPE, 2008, p. 199). 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 (ISPE, 2008, pp. 199-200).

GAMP5 foresees, that the test specifications and the test scripts can be recorded in a single document (ISPE, 2008, 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 (ISPE, 2008, p. 196).

As already mentioned, the test scripts are executed by the testers (ISPE, 2008, 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 screenshots (ISPE, 2008, 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 (ISPE, 2008, p. 196, p. 200). A test report includes again some metadata like who executed and who reviewed the testing, and information about the effected testing like a summary of the test results, a summary of test failures and conclusions (ISPE, 2008, p. 200).

Figure 4: Documents produced in the OQ main process

3.3.2 Incorporating the Quality Risk Assessment

Quality risk management is one of the five key concepts in GAMP5 (ISPE, 2008, p. 20). It is an iterative process that covers the entire life cycle of a computerised system (ISPE, 2008, p. 47). 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 (ISPE, 2008, p. 32). The goal of this concept is to focus validation efforts on critical points of the computerised system (ISPE, 2008, 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 (ISPE, 2008, p.107). This is in the responsibility of a team consisting of subject matter experts and key users (ISPE, 2008, p. 106), or if regulatory compliance is concerned, it is the quality unit (ISPE, 2008, p. 106). 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 (ISPE, 2008, p. 48, p. 50, p.106). Appropriate controls need then to be implemented (ISPE, 2008, p. 50), e.g. by the supplier as additional software functionalities (ISPE, 2008, p. 19).

As the goal of OQ is the documented verification of software functionalities (ISPE, 2008, p.38, p. 212), it is important to take the identified risks, their risk level and the defined control measures in respect to software functionalities into account (ISPE, 2008, p. 50). The specific level of test efforts should than be determined according to the risk level and the system impact (ISPE, 2008, p. 50) and

the controls themselves might be subject to the OQs (ISPE, 2008, p. 38). The risk assessment documentation needs finally be approved by the system owner and/or the quality unit (ISPE, 2008, p. 106).

3.3.3 Specification- and Test Management

Next to the quality risk assessment, there are other OQ supporting processes to be considered for a custom application (ISPE, 2008, p. 32). They include:

- **Change management process**[2]**:** "Change management procedures should also 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." (ISPE, 2008, 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 was performed and approved (ISPE, 2008, 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 (Figure 3), during which the approved tests will be re-run by documenting the new software version.
- **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" (ISPE, 2008, 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 (ISPE, 2008, 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 (ISPE, 2008, p. 33, pp.134-137). 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". (ISPE, 2008, p. 33). The process

[2] Typically the change control starts with the formal IQ execution (Evelyne Daniel, personal communication, April 25, 2020)

described by GAMP5 (ISPE, 2008, pp. 153-155) can be adapted in order to fit to the complexity of the project (ISPE, 2008, 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’ (ISPE, 2008, p. 154).
- 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 (ISPE, 2008, p.154, p. 60).
- The approval of a documents consists of a dated signature and reason for each approval signature, e.g. role (ISPE, 2008, p. 154). The document index and - history should be updated and the new status set, i.e. from ‘draft’ to ‘approved’ (ISPE, 2008, p.154). According to the role description in 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 (ISPE, 2008, p.58, p.196).
- The approved document is being issued by updating the document index (ISPE, 2008, p. 154). This could mean in respect to the OQ process, that the approved test scripts are handed over to the tester.
- Document changes need to be controlled by, for example, 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 (ISPE, 2008, 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 (ISPE, 2008, 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 (ISPE, 2008, p.155).

3.3.4 Exemplary OQ Process

Based on the descriptions made in the earlier chapters, an exemplary process was developed, which highlights the most important OQ relevant tasks, roles and documents (Figure 5). In view of the BDD process, the link to the functional specification process is also taken into account.

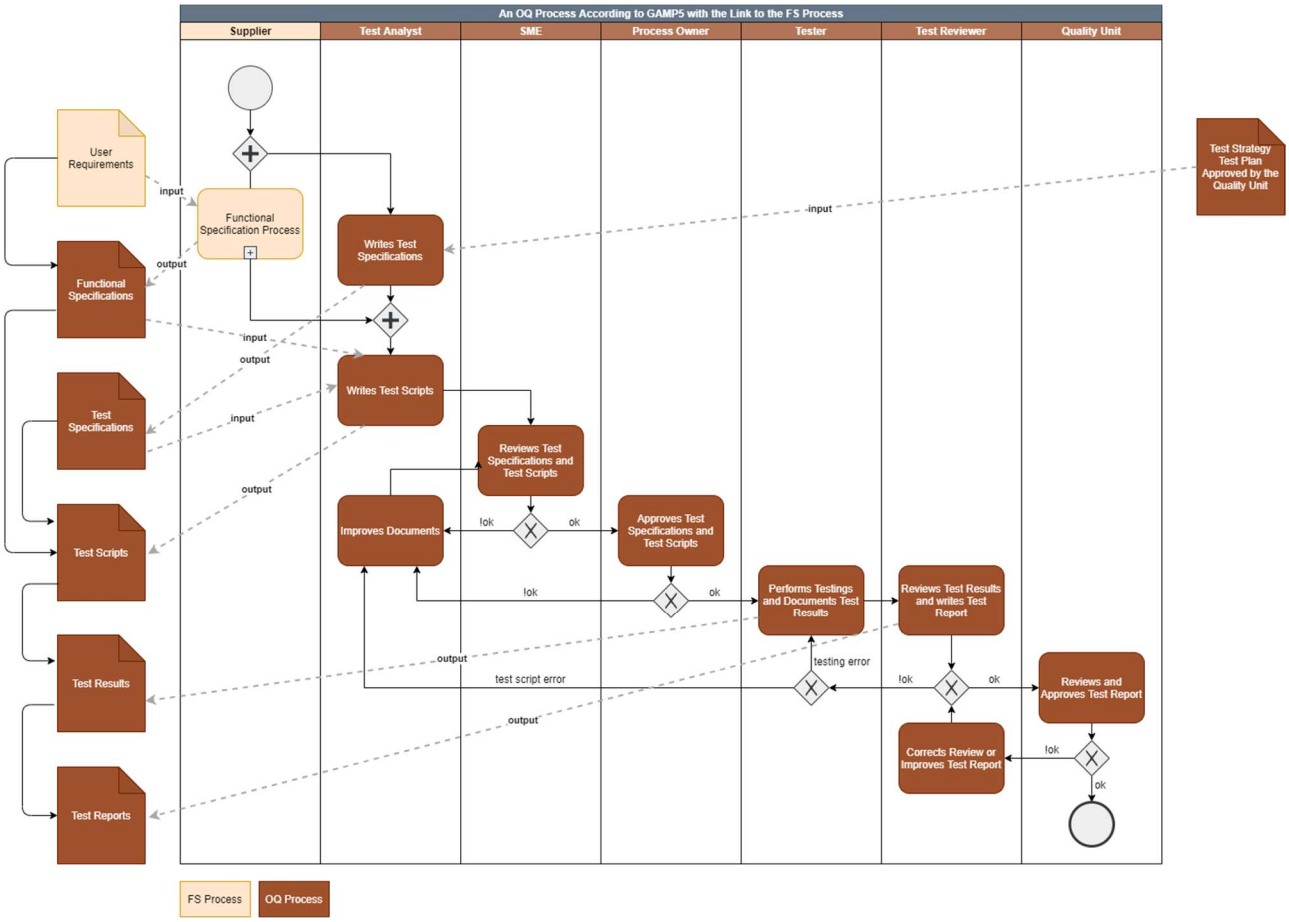


Figure 5: Exemplary OQ Process according to GAMP5

4 Behaviour Driven Development

Behaviour Driven Development (BDD) is a software engineering approach based on Test Driven Development (TDD) and developed by Dan North (2006). BDD focuses on a user-centred behaviour of the system to be implemented (North, 2006). To achieve this, the user requirements are written as executable test cases based on examples that are understandable for the business and that can be used by the developers for test automation to drive the implementation of the system in terms of TDD (North, 2006). The aim is to improve communication between business, testers and developers and to eliminate any misunderstandings as quickly as possible (North, 2006). Another important measure that BDD takes to promote good mutual understanding is that the user requirements/tests are based on concrete examples and are also written in this sense (North, 2006).

4.1 BDD a suitable Software Engineering Approach for Highly Regulated Environments

Gáspár Nagy and Seb Rose see great potential in the BDD for regulated environments such as the pharmaceutical industry (Nagy & Rose, 2018, pp. 72–74).

According to the two authors, BDD brings advantages especially with regard to the following increased requirements of regulated areas (Nagy & Rose, 2018, p. 72):

- Completeness and correctness of the specifications
- Test strategy and test coverage
- The required proof that the tests were performed in relation to a clearly defined version of the application.

Especially the following characteristics as listed by Gáspár Nagy and Seb Rose make BDD interesting for the regulated sector (Nagy & Rose, 2018, pp. 72–73):

- The specifications and the executable tests are merged into scenarios which are written in a language that can be understood by business.
- The scenarios will be developed in the form of illustrative examples to enable consistency and common understanding within stakeholders.
- The fact that the specifications in form of scenarios also serve as test scripts increases the consistency, and the traceability between test script and specifications becomes inherent.
- The test cases can be extended to achieve the required test coverage.
- The scenarios are versioned parallel to the application being developed
- BDD tools execute the scenarios directly as automated tests and generate a report on the test results, whereby this report

- documents the expected behaviour of the application,
- describes the tests that were executed to check the application
- provides the test evidence in relation to a given version of the application.

From the above statements, Gáspár Nagy and Seb Rose draw the following conclusions:" These statements about the BDD approach show that it is not only a good foundation for projects operating in regulated environments, but the living documentation satisfies the regulatory requirements better than classic processes. The improved efficiency comes from the fact that the scenarios are business readable, so you don't have to maintain a separate document for the tests. The code and tests become self-documenting" (Nagy & Rose, 2018, p. 73).

4.2 The Approach: An Overview

BDD has its origins in the agile world and way of thinking. It includes three central practices: Discovery, Formulation and Automation as shown in Figure 6 (Nagy & Rose, 2018, p. 20).

The goal of the first step 'Discovery' is to capture the user requirements of the system to be developed in its entirety (Nagy & Rose, 2018, p. 20). For this purpose, examples are recorded which illustrate a user requirement and business rules are defined which must be implemented in order for the user requirement to be considered as fully implemented (Nagy & Rose, 2018, p. 20 and pp. 36-39).

In the second step 'Formulation', the examples developed from the first step are documented in the form of scenarios in a 'Given'-'When'-'Then' structure and recorded on a so-called feature file (Nagy & Rose, 2018, p. 20 and p. 56).

These structured scenarios can then be automated in the third step 'Automation' so that the corresponding user requirements can be verified by a computer (Nagy & Rose, 2018, p. 20).

Figure 6: The three BDD practices (Nagy & Rose, 2018, p. 20)

Based on these three practices Gáspár Nagy and Seb Rose then build a detailed BDD process, which is summarized in Figure 7 (Nagy & Rose, 2018, pp. 56-61):

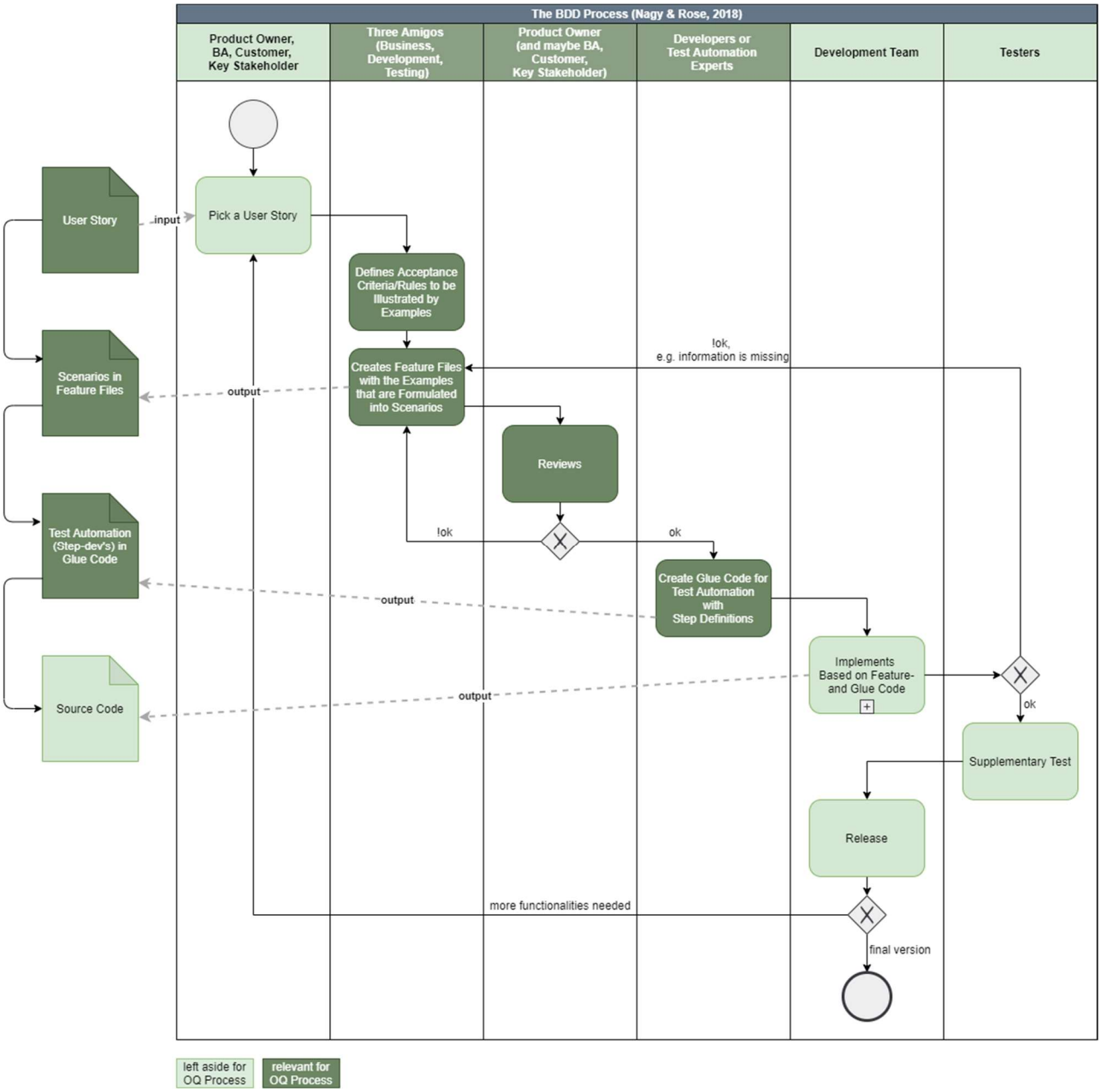


Figure 7: BDD Process according to Nagy & Rose (2018, pp. 56-61)

4.3 Defining user requirements as rules and with the help of examples

As described above, the first step is to examine and to discuss the user requirements to understand them in their full meaning (Figure 8).

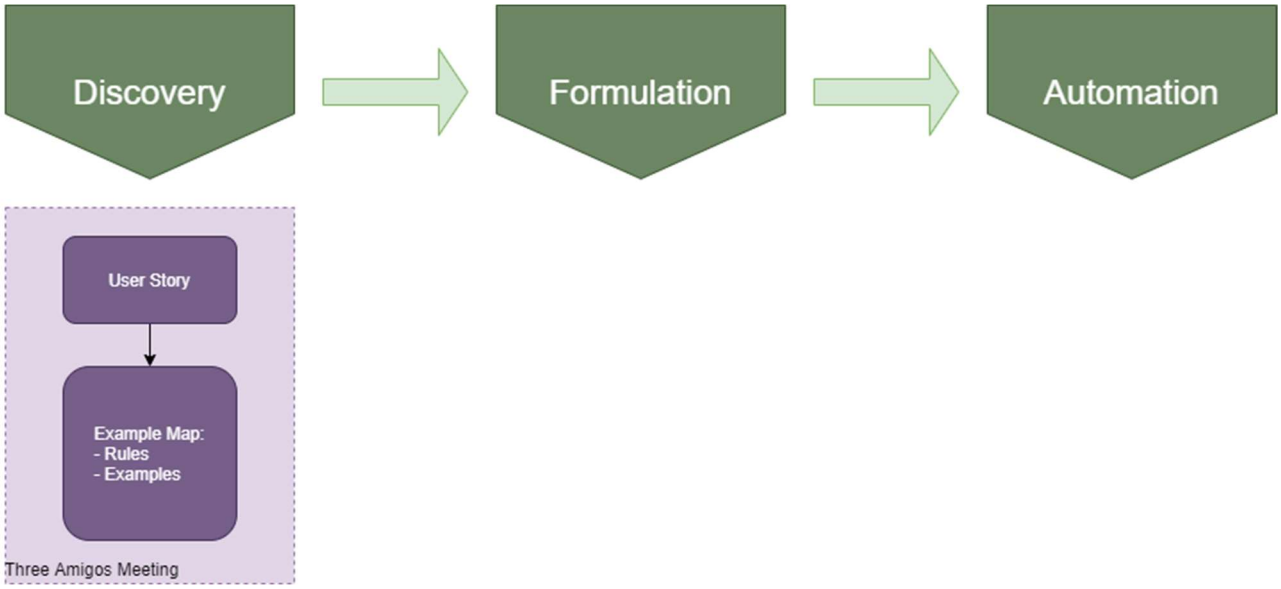


Figure 8: Activities within the BDD discovery step according to Nagy & Rose (2018).

This is done during the 'Three Amigos' meeting[3] (Nagy & Rose, 2018, p. 26, pp. 40-42; Agile Alliance, 2019). In the Three Amigos meeting, a user requirement is discussed and documented by means of the 'Example Mapping' method as shown in Figure 9 (Nagy & Rose, 2018, p. 26; Wynne, n.d.). As a result, various examples illustrating the user requirements and the underlying rules are available (Nagy & Rose, 2018, p. 35).

[3] According to Gáspár Nagy and Seb Rose this meeting is held under different names also including 'Discovery Workshop' or 'Specification Workshop' (Nagy & Rose, 2018, p. 26).

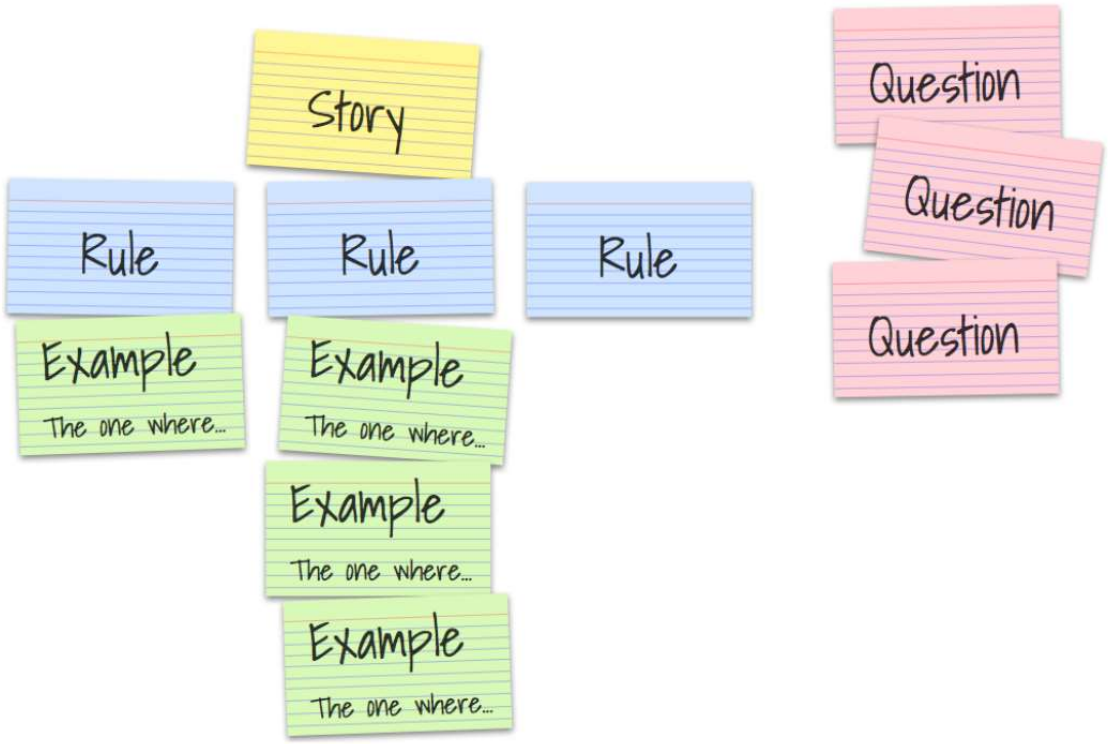


Figure 9: Example Map - structure and colour codes (Wynne, n.d.)

4.4 Writing Executable Specifications with Gherkin

As described above, the user requirements are specified using examples. These examples are then documented in a structured form in the 'Formulation' step. This further results in a document which is called a feature file (Figure 10) and which has to be included in the OQ process according to GAMP5.

To formulate the examples, a specifically developed language called Gherkin is used (Nagy & Rose, 2018, p. 20). Gherkin is a formalized language that allows to precisely specify the user requirements and at the same time to define the acceptance tests (Nicieja, 2018, pp.11-12). This language can also be understood by non-technical people and is based on the domain-specific vocabulary of the business (Nicieja, 2018, p.11).

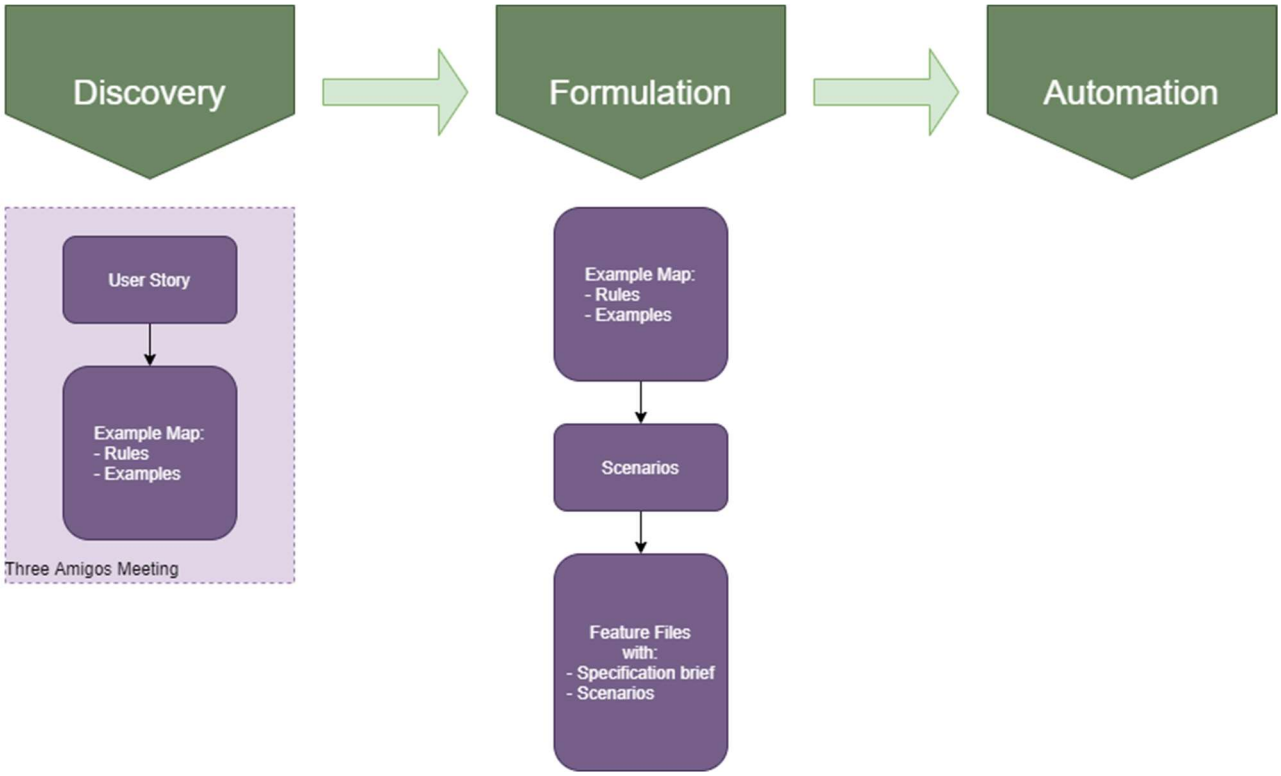


Figure 10: Activities within the BDD formulation step according to Nagy & Rose (2018)

4.4.1 The Scenario

The examples captured in the Three Amigos meeting are documented in scenarios with a 'Given'-'When'-'Then' structure using the Gherkin syntax (Nagy & Rose, 2018, p. 20 and p. 56; Nicieja, 2018, p.40). In the 'Given' part of the scenario, the context is defined and thus the prerequisites are determined (Nicieja, 2018, p.40). The 'When' part describes an event to be executed in the system while the 'Then' part contains the expected result (Nicieja, 2018, p.40).

Kamil Nicieja summarises this as follows:

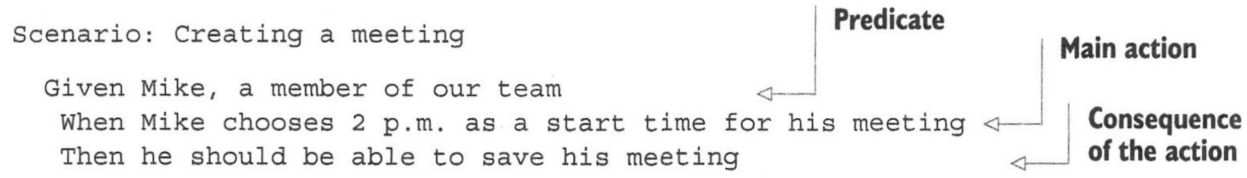


Figure 11: Example of a Scenario with the Given-When-Then structure from Kamil Nicieja (2018, p.43)

If one of these three parts consists of several sub-aspects, they can be extended accordingly using the keyword 'And' (Nicieja, 2018, p.44 and p53) as shown in Figure 12:

```
Given Mike, a member of our team
  And another event in Mike's calendar at 2 p.m.
 When Mike chooses 2 p.m. as a start time for his new meeting
 Then he should not be able to save his meeting
```

Figure 12: Example of a Scenario with the 'And' keyword from Kamil Nicieja (2018, p.53).

Each of these Given-When-Then statements is called a 'step' (Nicieja, 2018, p.44):

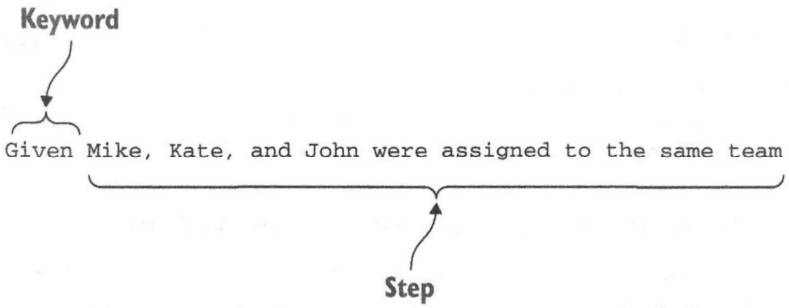


Figure 13: Example of a keyword and a step from Kamil Nicieja (2018, p.44)

4.4.2 Scenario outline

Besides scenarios, Gherkin also allows you to write scenario outlines (Nicieja, 2018, p.81). They allow similar scenarios to be grouped together and executed based on the same 'Given'-'When'-'Then' template (Nicieja, 2018, p.81). The differences between the summarised scenarios are mapped via parameterisation (Nicieja, 2018, pp.85-86).

This is then expressed as shown in Figure 14:

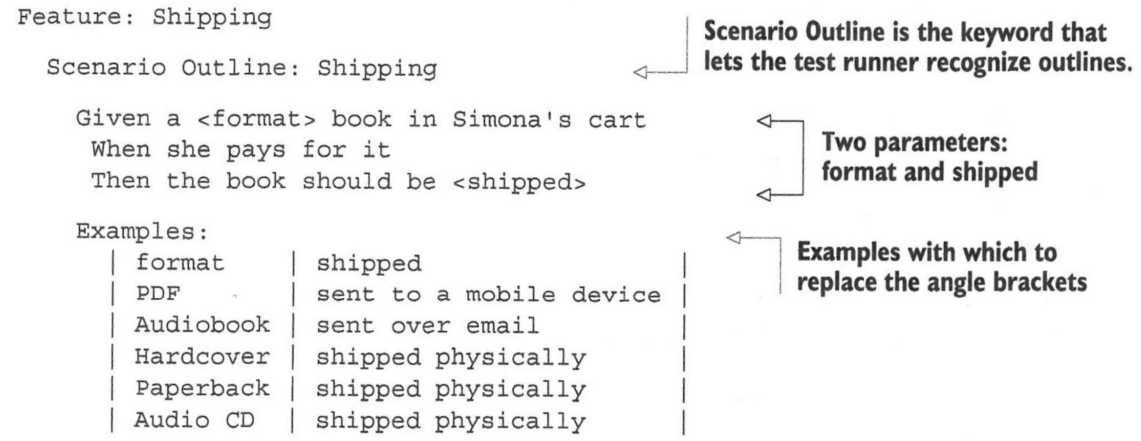


Figure 14: Example of a Scenario outline from Kamil Nicieja (Nicieja, 2018, p.86)

4.4.3 Feature File

Related scenarios are written down in a single document that is given the extension .feature and is accordingly called a feature file (Nicieja, 2018, p.34). A feature file contains all scenarios that describe a particular functionality of the application to be built (Nicieja, 2018, p.34). Each feature file begins with the word 'feature' and a unique name that describes the associated functionality (Nicieja, 2018, p.34, pp. 38-39). A very short feature file is shown in Figure 15 as a simple example.

```
Feature: Teams

    Scenario: Enforcing access rules

    For more information on availability, please look at the
    invite confirmation scenarios later in the specification.
    In short, an unavailable person who already confirmed RSVP
    can be invited to another event at the same time, but will
    be able to attend only one of them.

    Given Mike, Kate, and John were assigned to the same team
      And Ada was assigned to another team
     When Kate and Mike schedule a 1 hour long meeting at 4 p.m.
     Then John should see that Kate and Mike will be unavailable
      But Ada shouldn't be able to see Kate and Mike's meeting
```

Figure 15: Example of a feature file with one scenario from Kamil Nicieja (2018, p.34)

All feature files together and thus all executable Gherkin tests of an application are called 'specification suite' (Nicieja, 2018, p.34).

4.4.4 Specification brief and Scenario brief

In order to be able to record important information about a feature in the concerned feature file, such as the goal, affected stakeholders, description of the covered user requirements, etc., a free text can be entered just below the first line of a feature (Nicieja, 2018, pp.38-39). This text field is called 'Specification brief' (Nicieja, 2018, p.38) and looks like shown in the Figure 16.

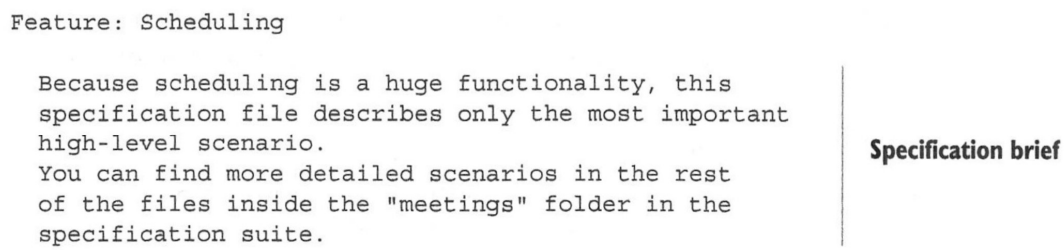


Figure 16: Example of a Specification brief from Kamil Nicieja (2018, p.39)

In the same sense, there is also a scenario brief defined, which allows information on individual scenarios to be recorded as illustrated in Figure 15 (Nicieja, 2018, p.40).

4.4.5 Summary

In the previous chapters some important aspects of Gherkin were briefly described. As seen, it allows a very concrete description of a functionality in a formal language, which can be understood without special Gherkin knowledge. It should be emphasized that, despite the formal language, there is a lot of room for customisation of these files, either in the context of using business-specific terms and phrases, or in terms of the possibility to include additional information in the Specification brief or Scenario brief sections.

4.5 Test Automation

The next step after the creation of the feature files, is their automated execution (see Figure 18). For this purpose, appropriate tools are required.

There are a lot of different tools that support BDD with its test automation part in various ways (SoftwareTestingHelp, 2020; Ketterlin Fisher, 2019). One of the most widely used BDD automation tools is Cucumber Open, and since it is also open source and freely available, this tool was chosen for this project (Ketterlin Fisher, 2019; SmartBear Software, 2020).

Cucumber works with Java and many other platforms, it allows test automation based on Gherkin, the established Junit Test automation framework and the use of the popular Selenium framework for GUI testing (Cucumber, n.d.-a; Tutorialspoint, n.d.-a, -b; Kazeeva, 2018; Flenner, 2020). As just described before, these tools are all recognised, widely used and established, which makes them also quite advantageous in the sense as they could be considered as GAMP5 category 1 testing tools as explained in chapter 3.1.5.

Cucumber is also quite flexible in the generation of reports as different formats can be used (Cucumber, n.d.-b). This has the advantage, that with the help of the Cucumber-Scenarioo-plugin (Hosbach, 2020), the Cucumber test reports can be viewed in Scenarioo, which allows automated documentation of UI tests (Scenarioo, n.d.-a).

Cucumber is a test execution engine that runs test code located in step definition or StepDefs in short (Nicieja, 2018, p.47). For each Gherkin step, a step definition is created with the test code to be executed (Nicieja, 2018, p.47). An example of this is provided in Figure 17. These step definitions act as glue code between the Gherkin steps and the actual test automation (Coveros, 2014; Rose, 2015).

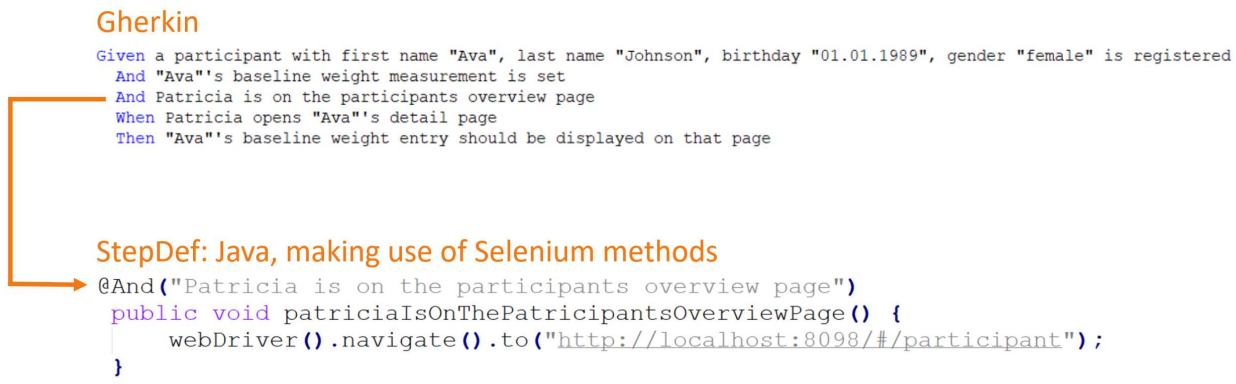


Figure 17: StepDef which automates a scenario step

Hence in order to be able to perform the actual test automation with the help of Cucumber, the glue code is developed based on the feature files. This enables Cucumber to test the target application and to create a test report (see Figure 18).

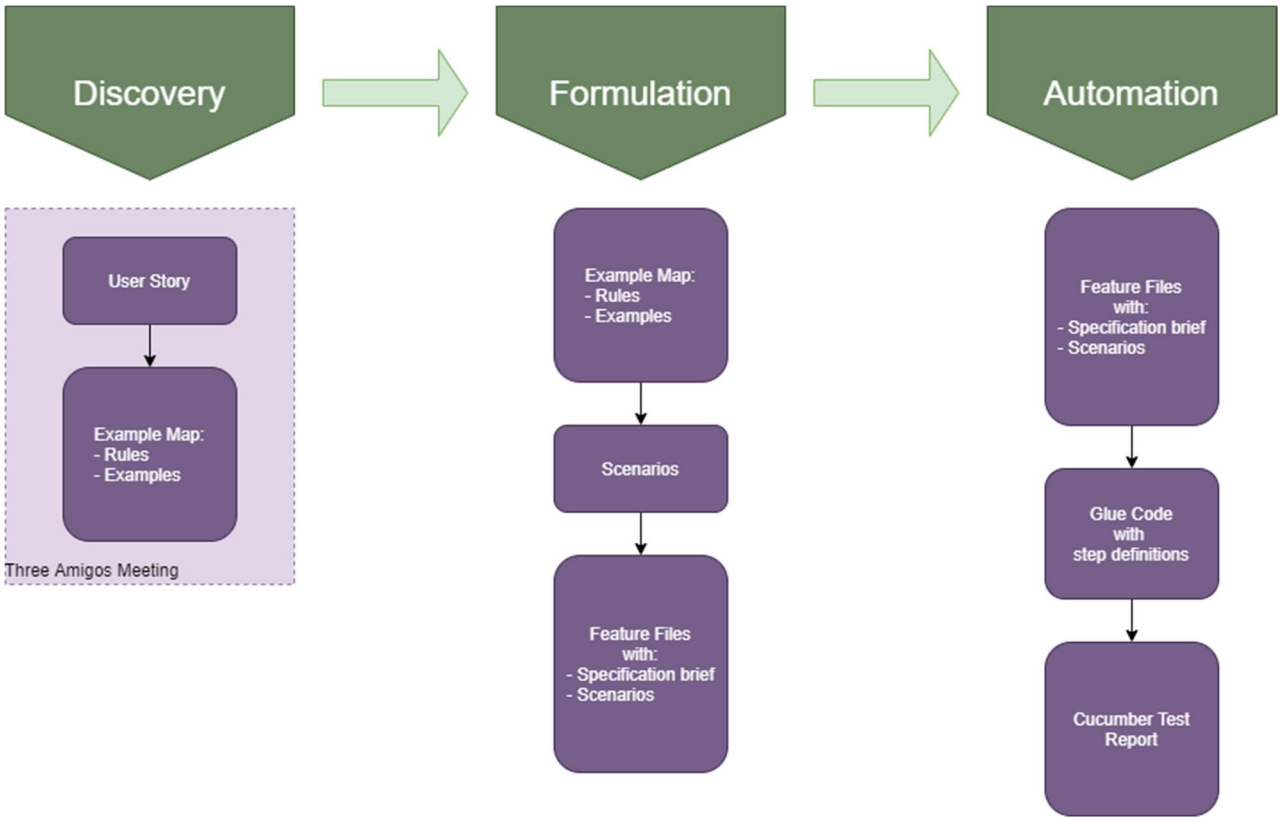


Figure 18: Activities within the BDD automation step according to Nagy & Rose, 2018

With these insights, it is now possible to investigate how the BDD practices can be integrated into the GAMP5 OQ process.

5 OQs with BDD

In order to obtain an initial assessment of whether and how BDD can be utilised to automate OQ testing, the OQ and the BDD processes (Figure 5 and Figure 7, respectively) were merged to examine the feasibility and possible consequences of such an automation.

5.1 The Combined Process

To define the combined process, the OQ process (Figure 5) was used as a starting point and wherever possible the BDD practices presented in chapter 4.2 were inserted.

In particular, the following two points had to be considered in more depth and solutions for their incorporation had to be found:

- In the OQ process the tests to be performed are defined based on the functional specifications whereas the BDD process starts with the user requirements to obtain the test cases.
- The actual tests are no longer performed by the tester but by the machine.

The other adaptations resulted from the two points mentioned above. The process created in this way is shown in the Figure 19.

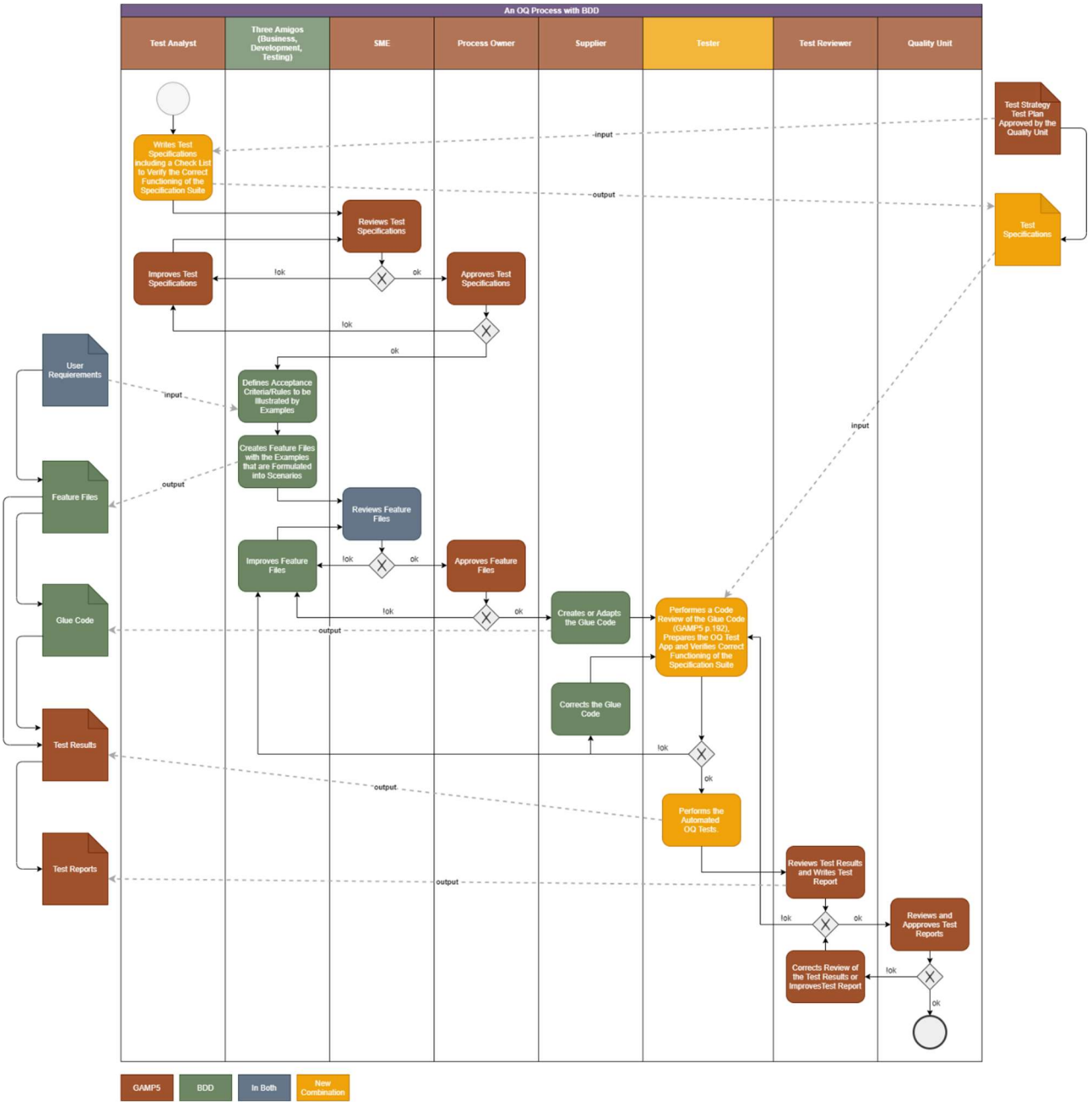


Figure 19: Process with integrated BDD practices

As intended by the OQ process, the process starts with the test analyst writing the test specifications for the planned OQ based on the test strategy, incorporating thereby the BDD practices and automated testing process.

This is followed by the preparation of the test scripts according to the BDD practices: In a Three Amigos Meetings the user requirements are examined in detail by a team[4] consisting of business, testing and development and are described with examples in order to create the scenarios documented in respective feature files. These can then be submitted to the quality assurance process in accordance with the GAMP5 OQ process. The functional specifications as well as the test scripts have thereby been reviewed and approved.

These are then the basis for the supplier to develop the application and the corresponding glue code. Ideally, if the supplier also follows BDD practices, they will use the feature files and their glue code to implement the application in TDD style.

Based on the feature files and the supplier's glue code, it is the responsibility of the tester to integrate them into the OQ test app and to verify the OQ Test App configured in this way for its proper functioning. In addition, further quality checks like glue code review or spot tests can be required, so that the OQs can be performed in a valid manner.

From the automated test process, the test results are generated, which must then be inspected and approved in a review process.

5.2 Discussion and Conclusions 'Combined OQ-BDD Process'

In principle it is possible to define an OQ process according to GAMP5 that includes the BDD Practices as shown in the chapter 5.1. It could also be imaginable to define another OQ-BDD process than the one shown in Figure 19 but with this process we have a proof of concept, that using BDD could be done in a way that is compatible with GAMP5.

Nevertheless, including BDD practices into the OQ process requires some fundamental changes compare to an OQ process with manual testing. The most important changes are discussed in the following chapters.

[4] Should consist on persons that do well know the practical side of the Requirements (User, Compliance, Regulatory Affairs), a tester, but not the persons having the tester role in the OQ Process, as they should be independent, and one or several representatives of the supplier.

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

GAMP5 defines that the document 'Functional Specification' should also describe the design constraints and the definitions of internal and external interfaces (ISPE, 2008, p. 175). GAMP5 further expects that only the actual functional specifications can be tested (ISPE, 2008, p. 175). Accordingly, only these can be considered within the scope of an OQ.

For this reason, it seemed sensible to detach the actual functional specification from the GAMP5 document 'Functional Specification' and transfer it to the OQ process.

This can also be well combined in terms of approval of the functional specifications and test scripts: As discussed in chapter 3.3.3, the roles of SME and Process Owner are suitable for reviewing and approving the test scripts. According to GAMP5, these roles are also predestined to approve the functional specifications (ISPE, 2008, p. 175, p 58, p. 60). This permits the process of approving the testable functional specifications and the creation of test scripts to be brought together.

As a consequence, the adapted GAMP5-functional specification process, which has the document 'Functional Specification' as output, would concentrate on the definition of the design constraints and the definitions of the interfaces, i.e. the actual non-functional requirements. How the non-functional requirements would then be integrated with the feature files into the 'Design-Specification' process (see Figure 3: Design- and verification process according to GAMP5) and further into the concrete implementation remains to be clarified, e.g. in a further project.

5.2.2 New Elements are Required

The tester role and the test specification must be redefined for the process shown above (Figure 19). Both elements already existed in the original OQ process, but their characteristics are clearly different with the new process:

- The tester now assumes a control function with regard to the correct execution of the tests by the machine. He no longer tests himself, or if he does, it would be with regard to supplementary quality assurance. Such tests would not be defined by a given test script. They would rather be built in as additional, unpredictable quality assurance based on the experience and competence of the tester (spot tests). To be able to fulfil these new tasks, a tester within the new process requires test management experience and experience with test automation and must therefore be able to read and understand code - i. e. he additionally needs technical knowledge.
- The test specifications define the procedure of the OQs to be performed based on the test plan. Accordingly, the new OQ-BDD process must be mapped and approved in these two

documents. The specific GAMP5 requirements for test automation must also be taken into account (compare 3.1.5).

5.2.3 Changes in the Documentation Set-Up

As already seen, the new OQ process is based directly on the user requirements to create the feature files. These feature files contain scenarios grouped according to the functionality of the application to be developed, whereby these scenarios represent both the functional specification and the test cases. This changes the basic structure of the document system from 'ordered by type' to 'ordered by content' as shown in Figure 20.

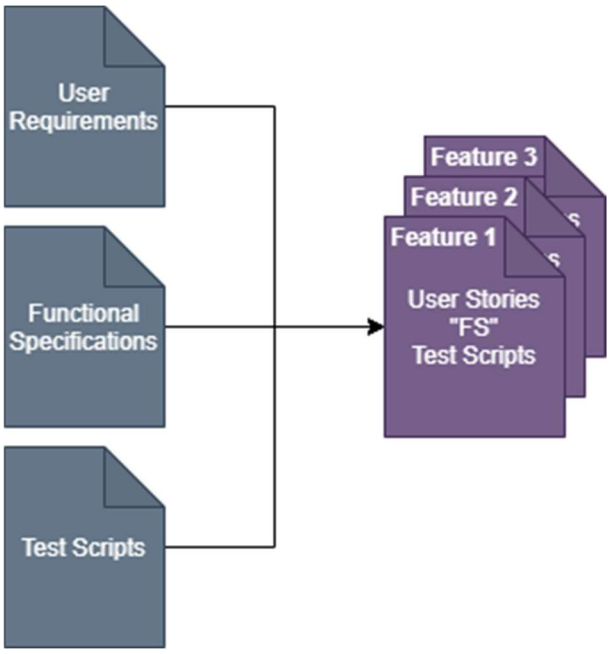


Figure 20: The documents are no longer structured according to the type 'User Requirement', 'Functional Specification', 'Test Scripts', but according to the functionality of the application, which is completely represented in each document

For the GAMP5 requirement of traceability this means a substantial simplification, since the feature files can be regarded as single source of truth from the user requirements, which are described or linked in the specification brief, via the functional specification to the test scripts (Hellesøy, 2015). In contrast, in the classic process, traceability from the user requirements via the functional specifications to the individual test cases must be guaranteed with an additional effort.

5.2.4 Conclusions after Analysis of the GAMP5 Requirement and the Processes

There was nothing that could be determined that would in principle prevent the use of BDD for the automated execution of OQ tests. On the contrary, some points even suggest that additional benefits could arise beyond test automation. These include in particular:

- Simplifications in traceability from the user requirements to the individual scenarios.
- Less redundancy in the individual processes, since the creation of functional specifications at least partially coincides with the OQ process
- Introduction of an additional quality check in the testing step, which strengthens the function of the tester and gives it more responsibility. However, the tester is also expected to have more technical knowledge about test automation.

Since so far nothing could be identified that could prevent the integration of BDD, a prototype is to be created in a next step to examine this new process in more detail with regard to the practical implementation.

6 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 standard operating procedures (SOPs) and a test plan. Where necessary for the understanding, aspects normally found in SOPs or in the test plan were 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 are described in chapter 6.3.

The prototype is based on example use cases and exemplary personas even though a scenario was used, that could be part of a clinical trial scenario especially in respect to the business requirements.

To better demonstrate the approach, a business app (JBA) was implemented. The JBA implements a minimum set of features without claiming any commercial or productive usage.

6.1 System Context and Application Design

The complete prototype system consists of three independent applications as also shown in Figure 21:

- The **Java Business Application (JBA)** was implemented as a Web application to simulate basic functionalities of a clinical trial management tool. The JBA represents the business application that needs to be validated.
- The **OQ Test App** was implemented to perform automated OQs on the JBA based on the feature files and the glue code. It generates the test results in XML and the screenshots in PNG format, so that they can be displayed in Scenarioo.
- **Scenarioo** is an open source software available and documented on its home page (Scenarioo, n.d.-a). Scenarioo is designed to support automated documentation of user interface tests. As it is not specifically designed for the usage in a regulated environment – e.g. in the pharmaceutical industry -, it is not yet validated.

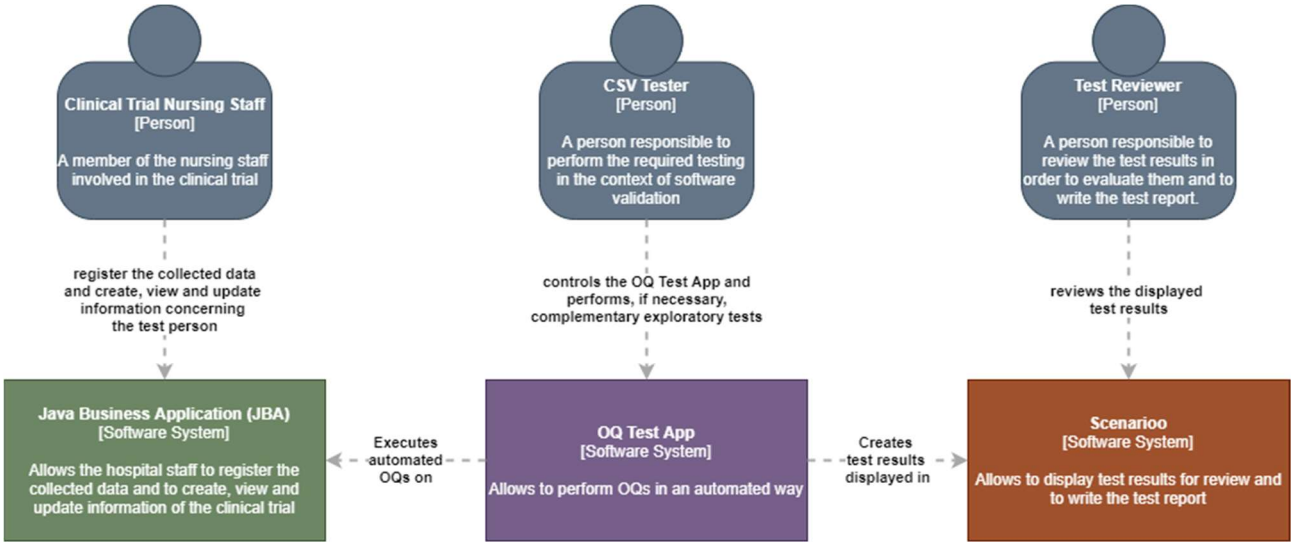


Figure 21: System context of the Prototype Design according to the C4 model

6.2 Architecture of the Apps used for Prototyping

6.2.1 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 on-premise application for a clinical trial management system in which the weight of the registered participant needs to be recorded (baseline weight measurement) before starting the test treatment.

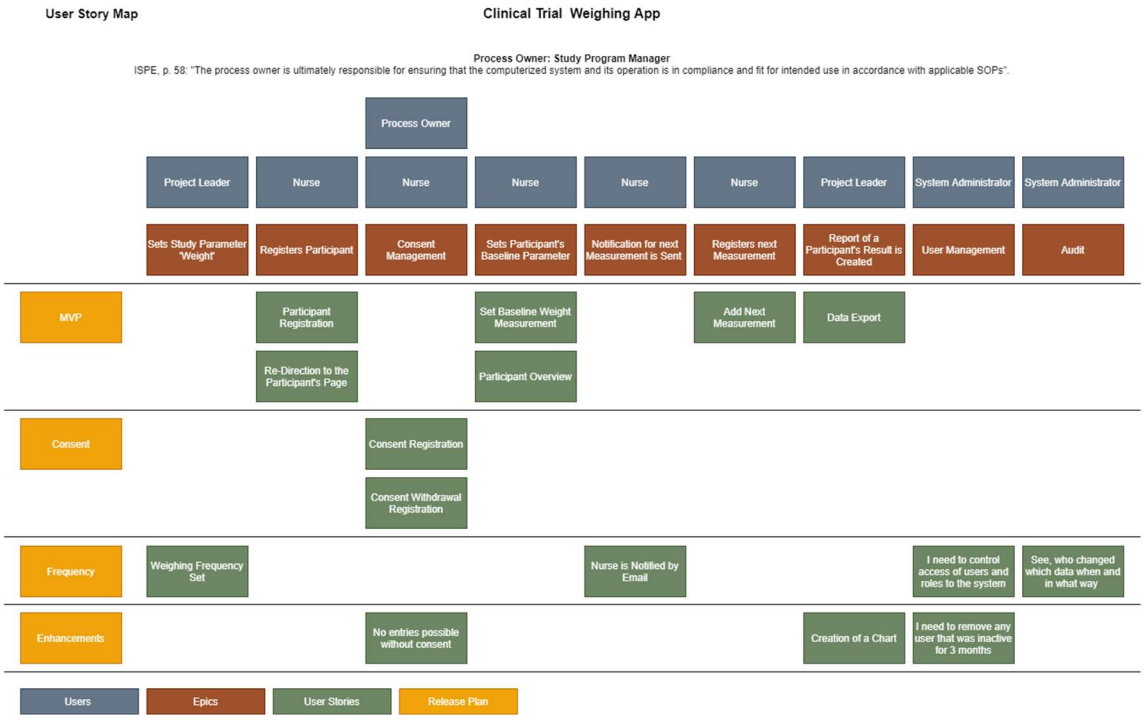


Figure 22: User story map as an overview of exemplary functionalities of the JBA

Generally, the business requirements of the JBA are not relevant to investigate BDD suitability for OQ Automation in GxP. Nevertheless, includes a feature to demonstrate how to deal with risks (registration of the baseline weight measurement) and another one to demonstrate how to deal with functionalities needed in order to be compliant (informed consent). Additionally, all implemented functionalities serve to test the overall GAMP5 OQ requirements for automated testing.

As shown in Figure 23: Container Diagram of JBA it is conceived as 3-tier application with a Vue.js single page Web frontend, a Spring Boot backend and a H2 in-memory database.

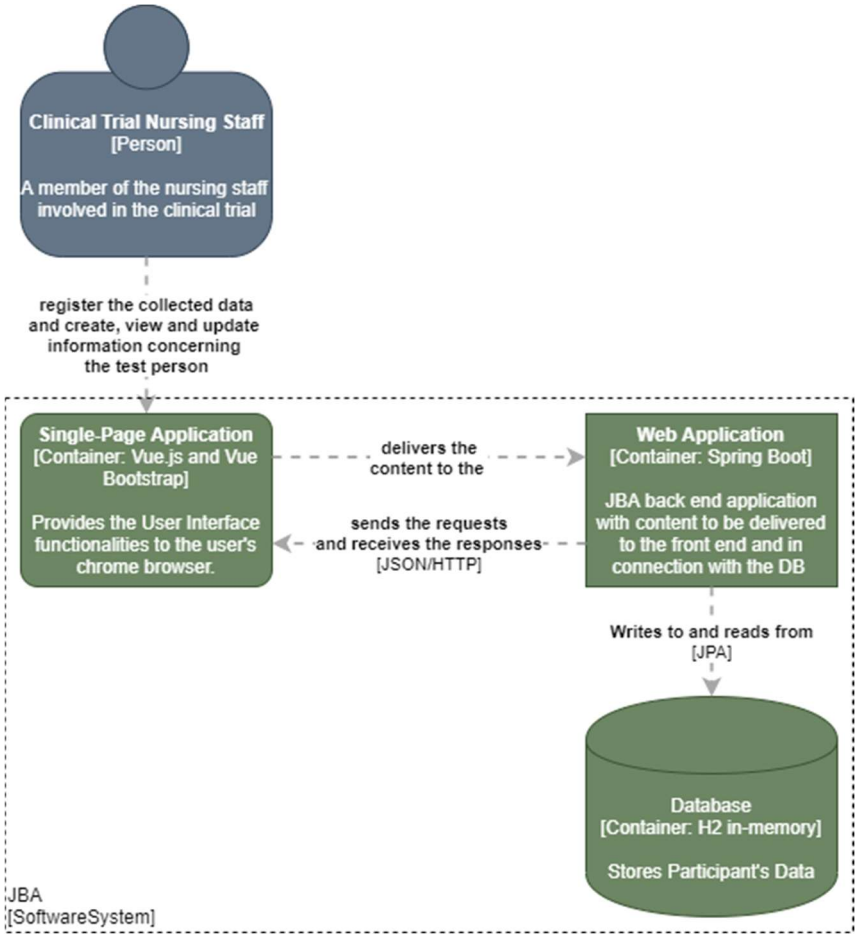


Figure 23: Container Diagram of JBA

JBA was set up as a Maven project because it provides very good dependency management. The dependencies with their respective version are fixed in the Maven POM files and additionally the package.json file for the frontend. All of them can be found in the Appendix II.

6.2.1.1 JBA Frontend

The JBA frontend was designed as a Vue.js single page application (Vue.js, n.d.). In addition, the user interface is based on the Bootstrap Vue components (BootstrapVue, n.d.).

The frontend module implements the user interface and is basically a data entry screen for the following data:

- Participant's data
- Baseline weight entry
- Informed consent

The JBA frontend includes four page types:

- The home page (Figure 24)
- The participant registration page (Figure 25)
- The participant overview page (Figure 26)
- Participant's detail pages (Figure 27)

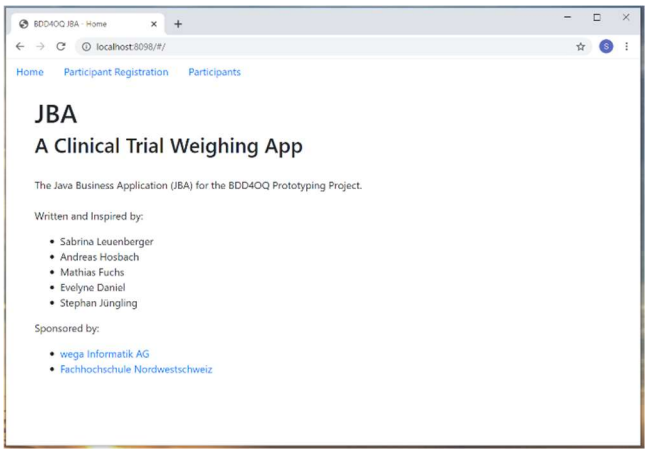


Figure 24: JBA Home Page

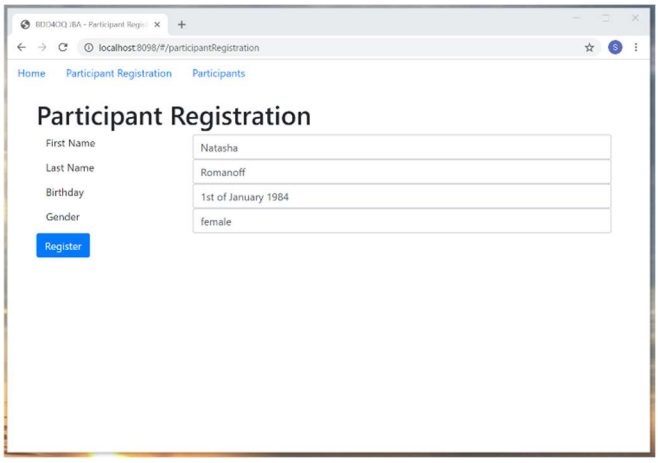


Figure 25: JBA participant registration

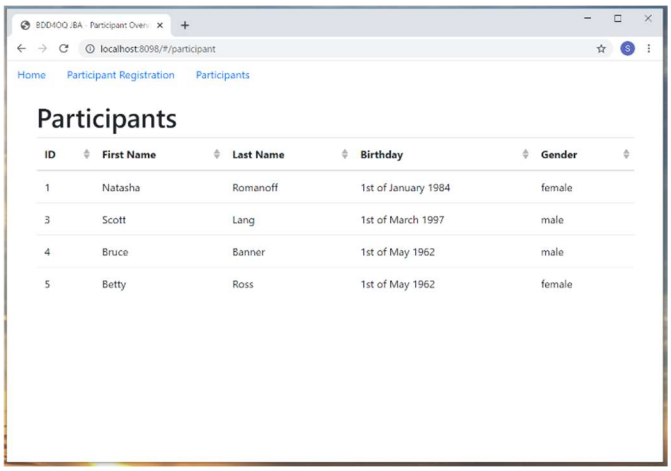


Figure 26: JBA participant overview

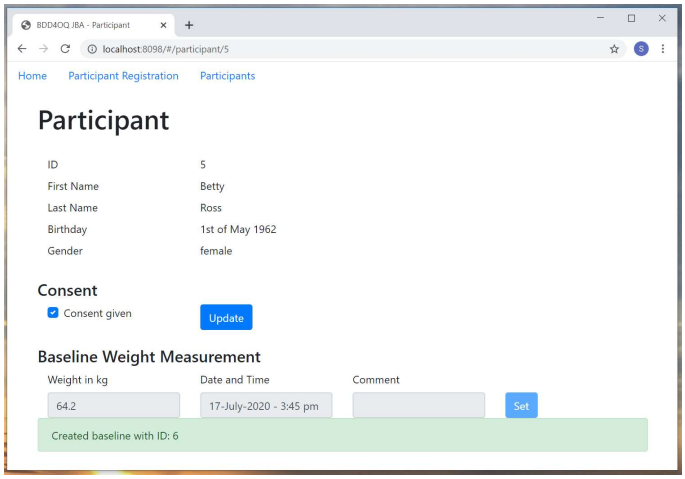


Figure 27: JBA participant's detail page

6.2.1.2 JBA Backend

The backend includes the business logic layer and the persistence layer of the JBA 3-tier architecture.

For the backend, Spring Boot was used as it is designed to create stand-alone Spring based applications and provides an embedded Apache Tomcat HTTP Web server environment (Spring, n.d.). The interface between the database and the business logic layer of JBA was managed by the Java Persistence API (JPA) integrated in the Spring Boot framework (Tyson, 2019; javaTpoint, n.d.). This allowed to fully implement the JBA Backend based on Spring Boot without need of caring about the database set-up.

As a database, the relational H2 Java SQL in-memory database was used (H2, n.d.). This choice was made because as an in-memory database it is not persistent. This s simplifies testing considerably, since it is empty again at every restart.

6.2.2 OQ Test App

The OQ Test App performs the OQ of the JBA. It is designed as an on-premise application under full control of the regulated company and to be run on the JBA test environment.

It is based on the JUnit Jupiter test framework with the cucumber test runner to perform the automated tests. The architectural design visualised using the C4 model (Brown, n.d.) is described in more detail in the two following chapters.

The version control of the different OQ Test App dependencies is guaranteed by a POM file (see Appendix III).

6.2.2.1 OQ Test App Container

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

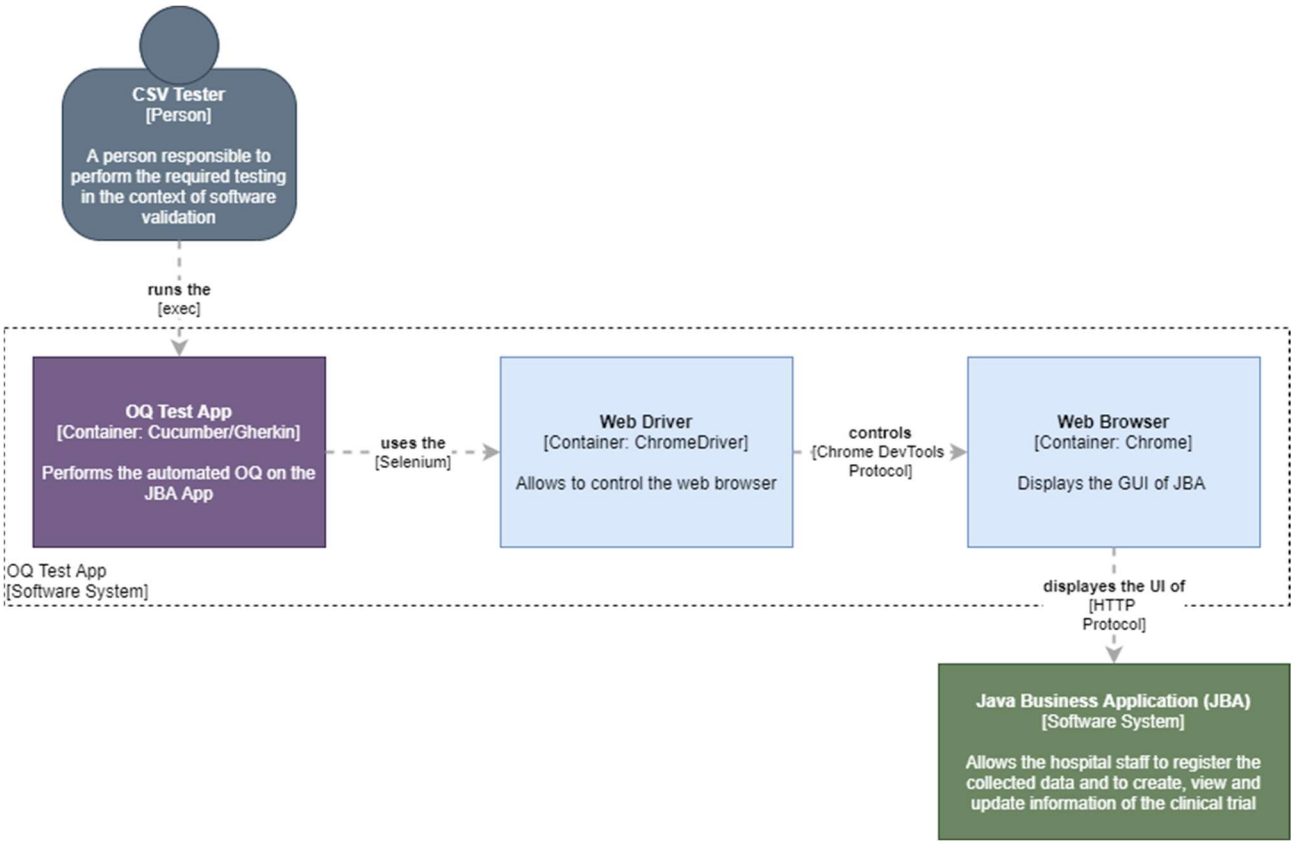


Figure 28: Container Diagram of the OQ Test App

In order to make it work, the Web driver corresponding to the respective Chrome version needed to be installed on-premise and referenced to in the OQ Test App. For the prototype Chrome version 83 and Chrome Driver version 83.0 were used (Chromium, n.d.; ChromeDriver Users, 2015).

6.2.2.2 OQ Test App Components

On a more detailed level, the components of the OQ Test App Container are shown and described in following figure (Anish, 2018; Tutorialspoint, n.d.-a, -b, -c; Coveros, 2014; Hosbach, 2020):

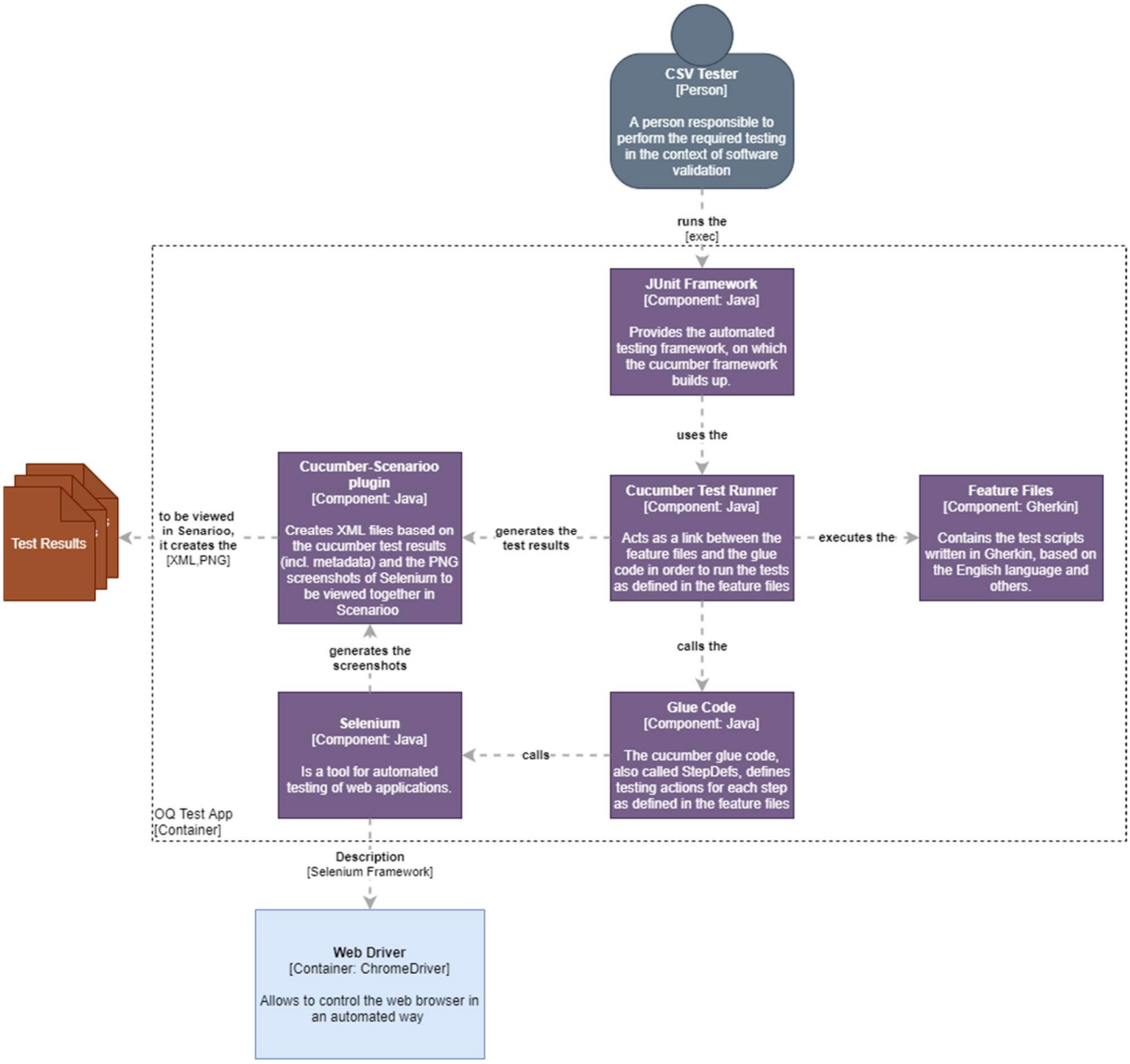


Figure 29: Component Diagram of the OQ Test App

6.2.3 Scenarioo

Scenarioo is used as an on-premise application. It is to be deployed together with the OQ Test App on the JBA test environment. The version 5.0.2.war was basically used out of the box. The only configuration performed, was the indication of the path to the test results in the application.properties file of Scenarioo:

```
scenarioo.data=C:\\Users\\ [......] \\target\\scenarioo
```

Figure 30: The only Scenarioo configuration

More information about this software can be found on the Scenarioo home page (Scenarioo, n.d.-a).

6.3 Suitability of the Selected Automation Tool Set for GxP Environments

As for any system in the regulated environment of the pharmaceutical industry, the test automation system consisting of the OQ Test App and Scenarioo would need to be validated (compare chapter 1.1). Therefore, the prerequisites for validation of this test automation systems were investigated to evaluate if such validation would principally be possible.

As already mentioned, the test automation system was designed to be 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 that system is the test environment of the JBA.

Prior to productive usage of test automation system, verification activities need to be performed dependent on the software category, the risk, the complexity and the novelty of the software (GAMP5 p.33-p.37; p.32). Additionally, to validate a computer system that consists of multiple components each of the components must be validated, therefore the two software of the system will be discussed in two separate chapters further below. But beforehand, the overall risk of the test automation system will be discussed in the next chapter, as this is the basis for every validation.

6.3.1 Risk-Assessment for the Test Automation System

GAMP5 defines validation as a “Risk based approach to Compliant GxP Computerized System” on its front page (ISPE, 2008, front page). This already implies that all validation activities are based on risk assessments at different levels as described earlier in chapter 3.3.2. The intended use of this test automation system is the automated OQ testing of the business application. The OQ Test App plays an important role within this system as it provides the test automation functionality that replaces the manual OQ test execution.

Two adverse effects of a poorly functioning test automation system could be identified that could pose a risk for the system and should be mitigated:

1. The OQ Test App does not perform all required functional tests as defined in the test script or Scenarioo does not display all test results. If this malfunction is not detected, the consequence would be, that the JBA will be deployed to production with an incomplete OQ.
2. The second risk that has to be mitigated is, that tests that should fail would not be detected as failures. In consequence they would be considered to have passed. In this case the JBA could be deployed to production with functionalities that do not fulfil the functional specification.

More adverse effects were found, but they would not lead to any risk. One such adverse effect was, for example, that a test would fail rather than pass. A risk due to this effect in terms of 'fitness for intended use' could not be inferred, as the consequence would be a re-testing of the relevant functionality of the JBA, with the result that this functionality would be correct, but the OQ Test App would contain a bug.

6.3.2 Single Tool Analysis for GxP Suitability of the Test Automation System

6.3.2.1 OQ Test App

The OQ Test App is a novel custom application and therefore a category 5 software according to GAMP5. This requires the highest level of validation efforts (ISPE, 2008, p.129, p. 130).

But having a closer look, the risk is assessable and straightforward (compare chapter 6.3.1), even though a full risk assessment has to be performed (ISPE, 2008, p. 107). This allows to address and mitigate it by defining a targeted verification strategy and specific test cases that should be easily manageable. Also in terms of complexity the OQ Test App should not be too difficult to validate, because it is a small and manageable application with a clearly defined goal and functionalities.

What might leverage these efforts even more is the fact, that at the component level, the software is mainly based on publicly available libraries as the:

- The JUnit framework
- The Cucumber test runner
- Gherkin
- And Selenium

These are all widely used and established testing tools (Ahmed, 2019; Cucumber, n.d.-a; Wikipedia, 2020; Unadkat & Krishnakumar, 2019) that can therefore be considered as category 1 software according to GAMP5 (ISPE, 2008, p.207).

The validation focus should therefore be more pronounced in regard of the feature files, the glue code and the Cucumber-Scenarioo-plugin.

As described in chapter 6.4, a specific strategy will be followed in respect of the feature files and the glue code to avoid revalidation of the OQ Test App for every JBA OQ execution.

Therefore, the last remaining component is the Cucumber-Scenarioo-plugin which needs to be considered as a category 5 software (ISPE, 2008, p.129, p. 130). The Cucumber-Scenarioo-plugin is responsible to make the Cucumber test report and the screenshots displayable in Scenarioo. This should also be quite easy to verify, i.e. that the required test metadata, the test results are correctly displayed in Scenarioo and that the screenshots are linked with the correct test step. If there is a malfunction in this plugin, it should be detectable with good test data and a good test script. To provide a reliable basis for such test data and test scripts, guidelines or SOPs should be established.

Next to these components, there is only very few own logic in the OQ Test App, as for example the authentication of the tester[5]. Therefore, also these functionalities should not pose any unmanageable barrier.

With this in mind and taking again into account the risks as described in the chapter 6.3.1, nothing could be determined, that would make it impossible to validate the OQ Test App and therefore its usage in a GxP environment.

6.3.2.2 Scenarioo

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

Scenarioo 5.0.2 is deployed as standard (out of the box) war (Web archive) file and 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 (Scenarioo, n.d.-b). Most importantly, no configuration was needed to ensure functionalities or interfaces next to the test result folder. Therefore, it is regarded as a category 3 software according to GAMP5 (ISPE, 2008, pp. 128-129).

As intended for a category 3 software, the risk assessment procedure can be simplified (GAMP5 page 108). The main risk would be, that Scenarioo displays not all results or that the results are wrongly displayed, these are risks that can be checked and excluded to a level to make them acceptable. Additionally, the software must not be considered as novel and also the complexity of the

[5] The authentication was only simulated in this prototype and not actually implemented.

software is manageable, as it is a small application with no dependencies that could not be handled and managed.

For category 3 software, the leverage of supplier involvement (ISPE, 2008, p.19) might be an aspect that could significantly simplify the validation. Unfortunately, no indication was found on the Scenarioo Web page, that Scenarioo would have been used already in a controlled environment of the pharmaceutical industry (Scenarioo, n.d.-a). Nevertheless, validation, especially with the supplier's support, should be feasible as it is a standard on-premise software with clearly defined functionalities to be verified and with only limited risk.

In conclusion, no reasons could be found, that would impede the potential validation of Scenarioo and therefore its usage in a GxP environment.

6.3.3 Analysis of the OQ Test App/Scenarioo System

As already described earlier (see chapter 6.1), 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 during the OQ Test App/Scenarioo integration is 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 chapter 6.3.2.1.

6.4 Updates of the OQ Test Automation System

In principle, any change in the software needs a re-validation of the new software version (ISPE, 2008, p. 30, Figure 4.1). For the components of the test automation system (the OQ Test App and Scenarioo), it can be decided by the regulated company, when to update and re-validate the applications as they are on-premise and in full control of the regulated company.

In respect to Scenarioo this means that the new version can be validated and installed whenever the improved Scenarioo version adds significant new value to the users which justifies the required validation efforts.

The situation of the OQ Test App is a bit different compared to the situation for Scenarioo: For each new OQ of the JBA, changes in the feature files and in the glue code, become necessary. This means that for each OQ the OQ Test App would need some re-validation.

A possible solution to avoid constant re-validation of the OQ Test App is therefore to 'outsource' the glue code and the feature files 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:

1. The remaining OQ Test App core consists on category 1 components, which are considered to be reliable. The validation efforts would need to be concentrated on the Cucumber-Scenarioo-plugin and on the small part of the OQ Test App's own logic as already described in chapter 6.3.2.1.
2. 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.
3. The correctness of the glue code and the feature files need to be verified in a different way than it would be done for software validation as it is not a software but just files.

For the further course of this project, it will be assumed, that the above described validation concept of the OQ Test App core and a separate process to verify the feature files and the glue code will be followed, even if for the prototype the feature files and the glue code were directly built into the OQ Test App. The verification process was therefore elaborated, as if the feature files and the glue code were imported into the validated and deployed OQ Test App core.

The rationale that was developed in order to allow the feature file and glue code verification is summarised in Figure 31:

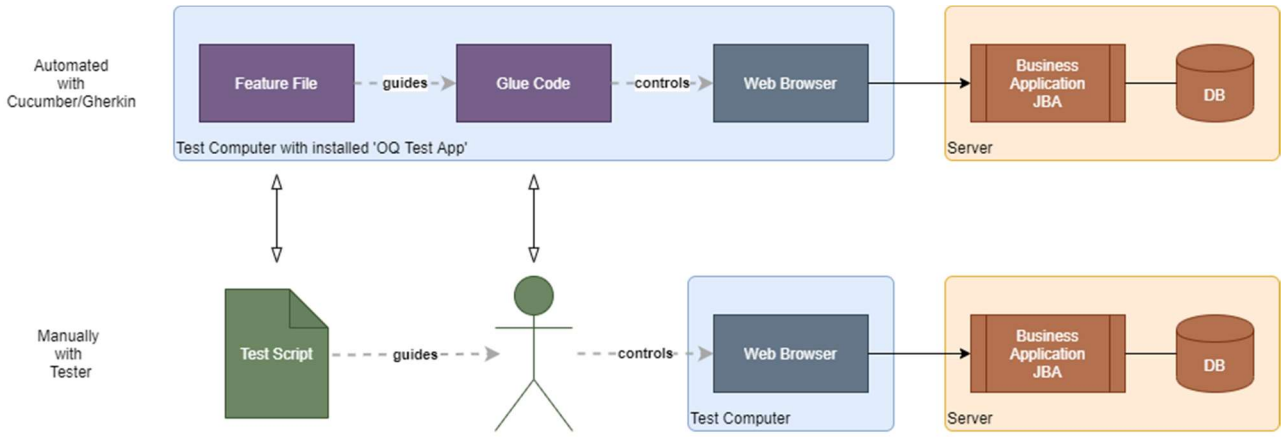


Figure 31: Glue code in analogy to a human tester, feature files are the test scripts

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.

To achieve this, the glue code uses methods provided by Selenium which in turn use the Web driver corresponding to the required 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

It should also be taken into account that the tester is prone to errors and the test scripts may contain errors. Therefore, the tester needs to take screenshots and write some descriptions as testing evidence. 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, which must reasonably be assumed to contain errors.

The same is true for the feature files: They might contain errors in the same way as classical test scripts might contain errors, which have then to be identified as test scrip errors during the test review process.

For this reason, Scenarioo plays an important role in displaying a test description and data for evidence on the test performance together with screenshots. This enables to verify the test results in the following test review. But it implies, that the tests and the screenshots have to be organised in a way, that the correct functioning of the glue code and the feature files can be reviewed.

The reviewability of the glue code can be achieved by checking for example following points during the code review:

- No empty functions in the glue code. This can be checked in two ways:
  - In the glue code review of the tester
  - By the test reviewer who has to verify, that each step took at least few milliseconds to perform.
- The glue code only performs actions on the UI in the same way a human tester would do. Selenium is a tool to automate UI testing. 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 (Hacker Girl, 2016). By only using Selenium to perform the UI actions, the human tester is simulated. The tester has therefore to make sure in the glue code review, that only Selenium functions are used for the actual test steps. For example while testing a Web application, there is the risk of making direct calls to the back-end. This would be easy to identify.

6.5 Specification/Formulation

In BDD, the specification (discovery) and formulation steps will turn the user stories into executable specifications (see chapters 4.3 and 4.4). To develop the JBA, we therefore proceeded according to these BDD practices, always taking into account the GAMP5 requirements.

6.5.1 From User Stories to Feature Files

The starting point for the JBA application of the prototype is based on a user story map that was created beforehand and adapted during the implementation process (see Figure 22). Based on this map, the user stories were formulated as in the Appendix XVIII.

As defined in the BDD process, a user story was then selected to be implemented next (see chapter 5.1). The selected user story was specified during a simulated Three Amigos Meeting using concrete examples and an example map:

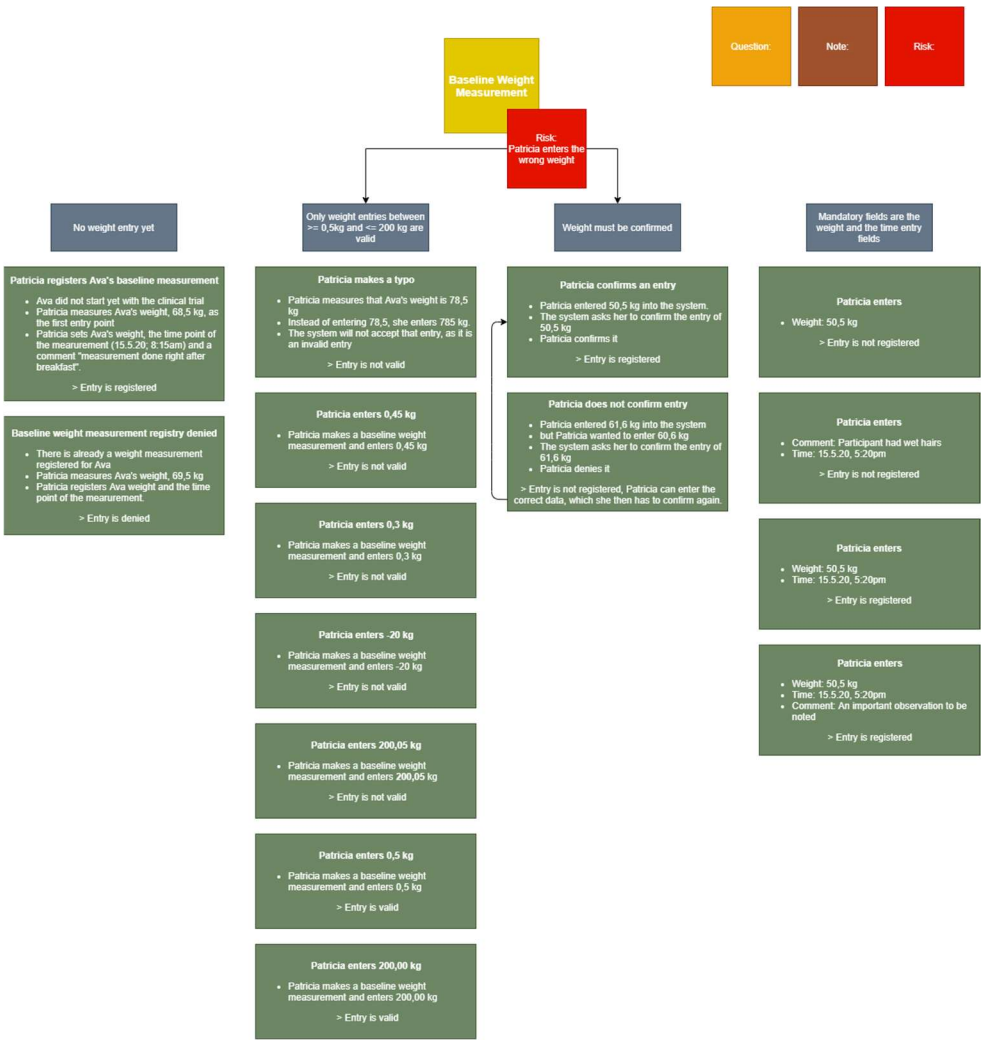


Figure 32: Example Map for the User Story ‘Set Baseline Weight Measurement’

Based on the example map, a first draft of the feature file was formulated and then adapted, reviewed and approved for OQ (Figure 19).

```
Feature: Setting the baseline weight measurement
  This specification describes how the nurse Patricia sets the baseline weight measurement of the participants.

  Covered Requirements:
  bddoq-26: Setting of the Participant's Baseline Weight Measurement

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-26	Sabrina Leuenberger	28-May-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	29-May-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	29-May-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	03-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	04-Jun-2020	wp
1.0.0.0	TS approved: Ready for OQ	Hank McKoy	04-Jun-2020	mh
  ```

  Size:
  10 active scenarios
  61 active steps


  Background:
    Given Patricia has the application open


  Scenario: Setting of the baseline weight measurement of a participant
  Patricia, the nurse, needs to set Ava's baseline measurement by entering the weight, the date and time and a comment.
  This represents the simplest happy path.

    Given a participant with first name "Ava", last name "Johnson", birthday "01.01.1989", gender "female" is registered
    And "Ava" has no baseline weight measurement entry yet
    And Patricia wants to set "Ava"'s baseline weight measurement
    When Patricia enters 68.5 kg in the weight field, "15.5.20, 8:15am" in the time field and "measurement done right after breakfast" in the comment field
    And she saves these entries
    Then "Ava"'s baseline weight entry should be found in the system

  Scenario: Displaying baseline weight measurement of a participant
  Patricia needs to see the baseline weight measurement entry of a participant once it is set.

    Given a participant with first name "Ava", last name "Johnson", birthday "01.01.1989", gender "female" is registered
    And "Ava"'s baseline weight measurement is set
    And Patricia is on the participants overview page
    When Patricia opens "Ava"'s detail page
    Then "Ava"'s baseline weight entry should be displayed on that page

  @Ignore
  Scenario: Denial of setting the baseline weight measurement when it already exists.

    Given Ava with first name "Ava", last name "Johnson", birthday "01.01.1989", gender "female" is registered
    And Ava has already a baseline measurement
    When Patricia wants to register Ava's baseline weight measurement
    Then she should not be able to register a new baseline measurement

  Scenario Outline: Allowed weight entry values: <weight>
  To minimise the risk of wrong entries, only entries between >=0.5 and <=200 are valid

    Given a participant with first name <first_name>, last name <last_name>, birthday "21.09.2014", gender "male" is registered
    And <first_name> has no baseline weight measurement entry yet
    And Patricia wants to set <first_name>'s baseline weight measurement
    When Patricia enters <weight> kg and any valid date time
    Then she can set the baseline weight measurement
    Examples:
      | first_name | last_name | weight |
      | "Alec"     | "Turner"  | 0.5    |
      | "Elec"     | "Turner"  | 200.0  |
      | "Ilex"     | "Turner"  | 12.8   |

  Scenario Outline: Forbidden weight entry values: <weight>
  To minimise the risk of wrong entries, values are invalid when they are not in the range between >=0.5 and <=200

    Given a participant with first name "Eric", last name "Turner", birthday "21.09.2014", gender "male" is registered
    And "Eric" has no baseline weight measurement entry yet
    And Patricia wants to set "Eric"'s baseline weight measurement
    When Patricia enters <weight> kg and any valid date time
    Then she cannot set the baseline weight measurement
    Examples:
      | weight |
      | 785    |
      | 0.45   |
      | 0.3    |
      | -20.0  |
      | 200.05 |
```

Figure 33: Example of a JBA feature file, that is approved for OQ

The complete set of feature files can be found in the Appendices IV-VII.

6.5.2 Traceability

As already described in the introduction to GAMP5, the traceability of the user requirements to the functional specifications and the OQ tests must be assured (see chapter 3.3.3).

The feature files are ideally suited for this: The specification brief section of the feature files describes the user requirements covered by the file. The user requirements can either be inserted directly into the specification brief and managed there, or a unique link to the described user requirement can be inserted. Within the prototype the user requirement was inserted as a link via the unique ID of the Jira user story in the specification brief - section (see Figure 34).

The functional specifications are described as scenarios and since they also constitute executable test scripts, the traceability is already given by itself.

In summary, it can be concluded that traceability can be easily ensured, as a feature file is structured in such a way that it contains all elements that need to be traced.

Figure 34: Link between the description of the user requirement and the executable functional specifications on the feature file

6.5.3 Risk Assessment

As already described in various previous chapters, the GAMP5 concept is very much based on the management of risks. The risks are recorded and evaluated at different levels and in all parts of the process. Where necessary, measures must be taken to mitigate them. To meet this GAMP5 requirement, the example mapping was extended by the card type 'Risk' and the colour code was adapted correspondingly: The 'Risk' card was assigned the colour 'red' and the 'Question' card was reassigned the colour orange. In this way, the discussion about risks and how they can be mitigated should be promoted and systematically included in the Three Amigos Meeting. An example of how this was realised for the prototype can be seen in the Figure 35. The examples and rules, which were recorded in view of mitigating a risk in the Three Amigos Meeting, were converted into specific scenarios. The scenario brief was then used to add information about this risk mitigation measure.

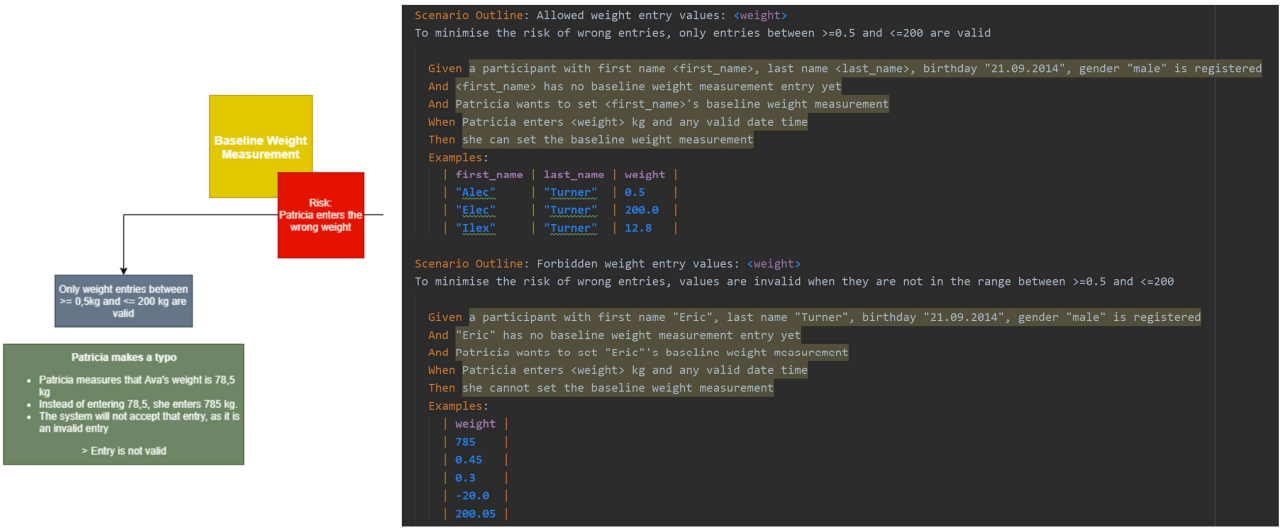
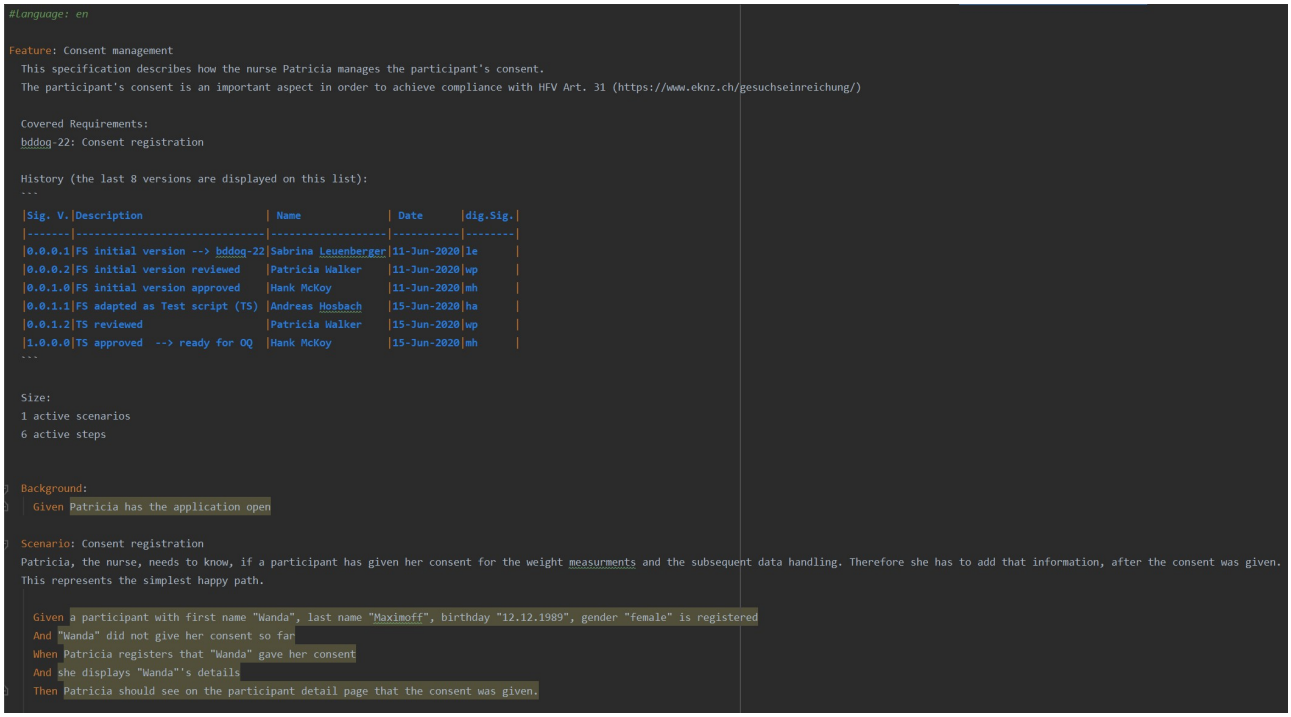


Figure 35: Example of how to deal with GAMP5 risk management requirements in BDD

6.5.4 Compliance

As seen in the introduction (chapter 1.1), the purpose of validation is to demonstrate that the software is fit for its intended use and that it is compliant to applicable regulations.

One such compliance requirement is for example the 'Informed Consent' (FDA, 2018). In order not to lose the connection between legal requirements and their representation in the system, the prototype was referred to the underlying legal basis in the 'Specification brief' section of the feature file concerned (see Figure 36).



```
#Language: en

Feature: Consent management
  This specification describes how the nurse Patricia manages the participant's consent.
  The participant's consent is an important aspect in order to achieve compliance with HFV Art. 31 (https://www.eknz.ch/gesuchseinreichung/)

  Covered Requirements:
  bddoq-22: Consent registration

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-22	Sabrina Leuenberger	11-Jun-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	11-Jun-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	11-Jun-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	15-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	15-Jun-2020	wp
1.0.0.0	TS approved --> ready for OQ	Hank McKoy	15-Jun-2020	mh
  ```

  Size:
  1 active scenarios
  6 active steps


  Background:
    Given Patricia has the application open

  Scenario: Consent registration
  Patricia, the nurse, needs to know, if a participant has given her consent for the weight measurments and the subsequent data handling. Therefore she has to add that information, after the consent was given.
  This represents the simplest happy path.

    Given a participant with first name "Wanda", last name "Maximoff", birthday "12.12.1989", gender "female" is registered
    And "Wanda" did not give her consent so far
    When Patricia registers that "Wanda" gave her consent
    And she displays "Wanda"'s details
    Then Patricia should see on the participant detail page that the consent was given.
```

Figure 36: Feature file with a reference in the specification brief to the underlying legal basis

6.5.5 Approval of the feature files

As already described more in detail in chapter 3.3.3, OQ documents need to be approved. This applies in particular to the feature files. From the experience of creating the prototype, it must be expected that the feature file, which was created in the 'Formulation' step, has to be adapted at least formally for the 'Automation' step.

As shown in the OQ process and the combined process, the SME is responsible for the review and the process owner for the approval of the functional specifications as well as the test scripts (chapter 5.2.1, Figure 5 and Figure 19). Therefore, it is also them who should approve the feature files.

The following concept was thus applied to the prototype: First approval step of the feature files as functional specifications after formulation and second approval step of the feature files as test script for the pending OQs after implementation of the automation[6]. This was recorded accordingly in the document history in the specification brief of the feature file.

[6] According to an analysis done by wega there is more freedom, in regulations like Annexe 11, compared to GAMP5, as to when which document has to be approved. For example, it can be assumed that OQ test scripts can also be approved only after execution of the OQs, e.g. together with the test report (Evelyne Daniel, personal communication, May 27, 2020). Since this project strictly adheres to GAMP5, no use was made of that freedom. In this sense, however, it can be assumed that the proposed approval process could be further simplified, e.g. by a one-time release of the feature files shortly before the actual OQ implementation (Evelyne Daniel, personal communication, May 27, 2020). A further possibility to simplify would be that the feature files are only approved once as a functional specification and that the test review serves additionally to approve the feature files as test scripts retrospectively.

```
Feature: Setting the baseline weight measurement
  This specification describes how the nurse Patricia sets the baseline weight measurement of the participants.

  Covered Requirements:
  bddoq-26: Setting of the Participant's Baseline Weight Measurement

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-26	Sabrina Leuenberger	28-May-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	29-May-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	29-May-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	03-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	04-Jun-2020	wp
1.0.0.0	TS approved: Ready for OQ	Hank McKoy	04-Jun-2020	mh
  ```
```

Figure 37: Feature file approval and document history

6.6 Test Automation

In order to achieve test automation, StepDefs were written based on the feature files in order to store an automated action for each step of the scenarios. All StepDefs together are called glue code (compare chapter 4.5).

Next to the StepDefs, the glue code also contains hooks to execute functions that are not directly related to a scenario steps, as shown as example in Figure 40.

To execute the feature files, a Cucumber test runner is required which was configured as shown in the Figure 38:

```
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/main/resources/jba_feature_files",
        glue = "oq_glue_code",
        strict = true,
        plugin = "com.github.andreashosbach.cucumber_scenarioo_plugin.CucumberScenariooPlugin:src/main/resources/cucumber_scenarioo_config.json",
        tags = "not @Ignore"
)
public class OQRunner {
}
```

Figure 38: Configuration of the Cucumber Test Runner

6.6.1 Glue Code

For the implementation of the glue code some rules were established. These rules are based on findings described above (chapter 6.4) and on experiences made during the implementation of the prototype. This should ensure consistency with the OQ process rational and reviewability of the test results. Compliance with the defined rules must be ensured by the tester during the code review.

In practice, such rules would be recorded in a coding guideline (see Appendix XIV).

The following points could be established, which were taken into account within the concept elaborated here:

- Control of the test automation only via the GUI: As stated in the discussion on the updates of the OQ Test Automation System and in this context the handling of the glue code (see chapter 6.4), the test execution should only be done via the GUI in analogy to the human tester. Steps which are upstream or downstream of the actual test to create the right test conditions, e.g. creating/deleting test data can access other interfaces, e.g. Rest API.
- No empty methods in StepDefs: Steps based on StepDefs with empty methods will result in a 'success', although no actual test has taken place. Therefore, it is important to prevent StepDefs from consisting of empty methods. This can be achieved via code review, and as these steps do not need any execution time, they also can be detected during the test review:

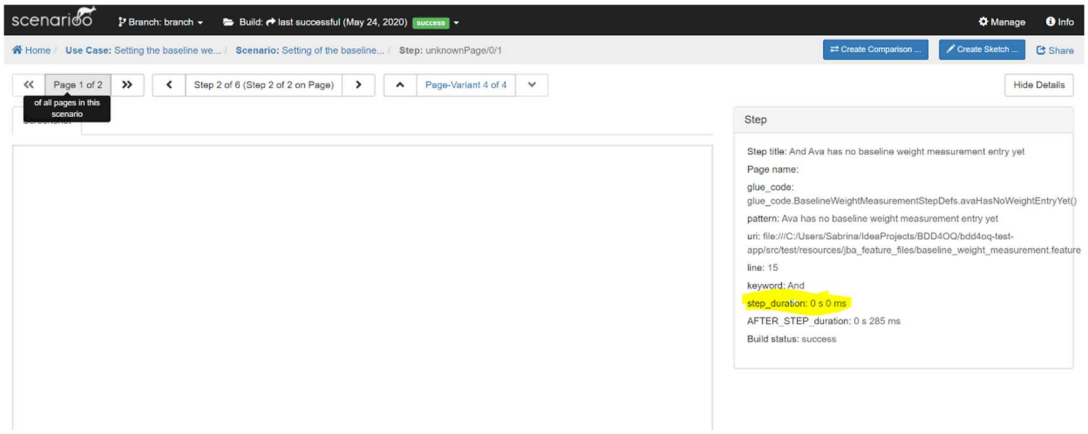


Figure 39: Scenarioo visualization of a step whose StepDef contained only an empty method

- Screenshots: After each step, the Cucumber test runner uses Selenium and the Cucumber-Scenarioo-plugin to perform a hook that takes a screenshot of the currently displayed JBA page:

```
@AfterStep
public void afterStep() {
    TakesScreenshot scrShot = ((TakesScreenshot) (webDriver()));
    String title = webDriver().getTitle();
    Screenshot.save(title.isEmpty() ? "Unavailable" : title, scrShot.getScreenshotAs(OutputType.BYTES));
}
```

Figure 40: Hook, which is responsible for taking and saving the screenshots

In order to write the glue code more clearly and maintainable, it turned out to be helpful if the GUI elements of the JBA were given a descriptive and unique ID. This allowed the elements to be addressed by Selenium via this ID. If, on the other hand, the elements are accessed via the document object model (DOM), the correct element may no longer be found, or the wrong element may be accessed if the DOM has changed due to a change on the Web page.

The glue code implemented in the course of this project can be found in the Appendices VIII-XIII.

6.6.2 Test Results as Cucumber Reports

The test results were generated by the Cucumber test runner in combination with the Cucumber-Scenarioo-plugin as XML reports and linked with the screenshots. To make these test results accessible, they were stored in a folder that was automatically created specifically for this purpose so that this information could be displayed by Scenarioo.

6.7 Test Review

As explained, a test review must take place after the test execution to meet the GAMP5 requirements. The purpose of the test review is to check the test results and to uncover any errors in the JBA, the feature files or the glue code. Based on the test review, the next steps are determined in a risk-based manner. In the ideal case, the OQ is passed, the OQ process can be completed and the PQ process can be started.

6.7.1 Review in Scenarioo

To enable the test review, Scenarioo was given the path where the folders with the test results are stored. This gives Scenarioo access to the test results and enables it to display them in its user interface:

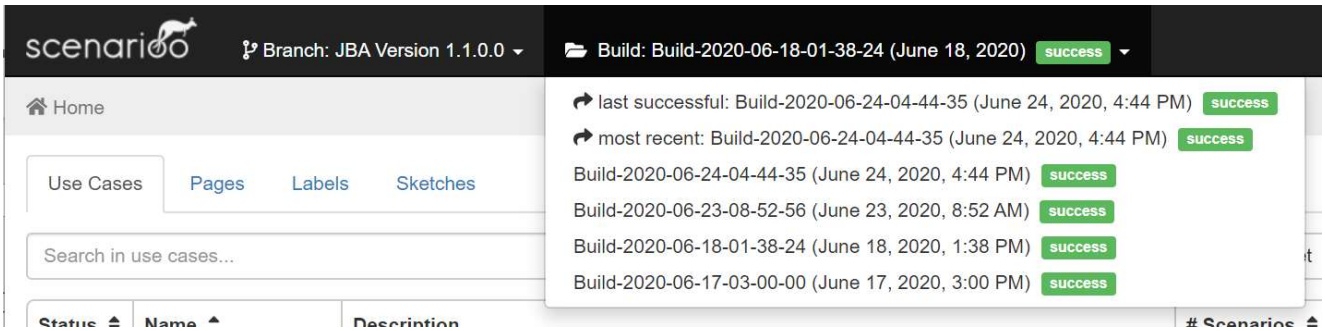


Figure 41: Different runs that can be viewed in Scenarioo

Scenarioo presents the test results on four levels:

1. The overview level on which the features (called 'use cases' in Scenarioo) and the final result of the tests for each feature is displayed:

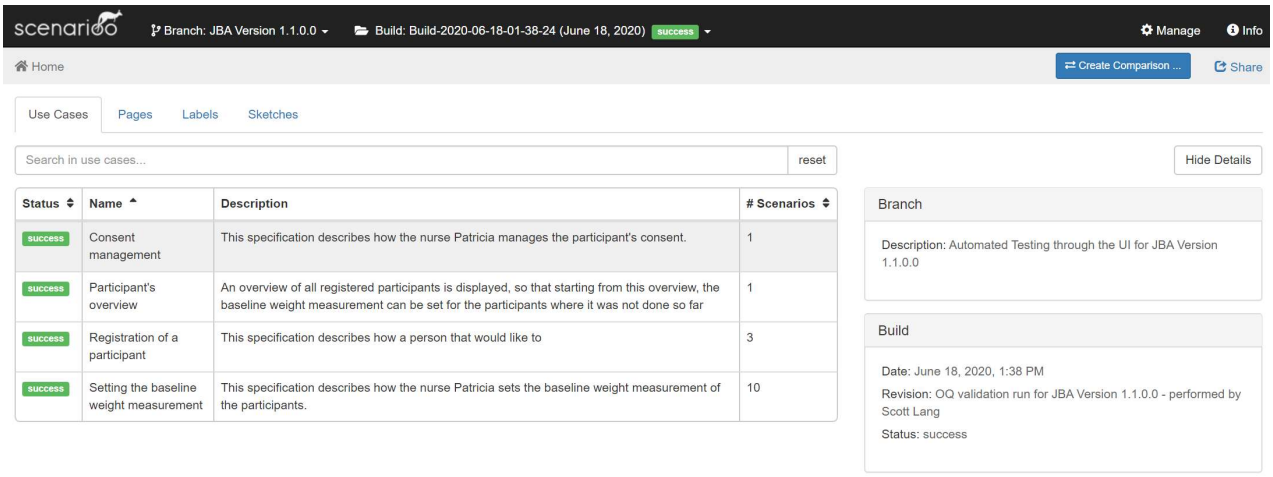


Figure 42: Feature overview in Scenarioo

At this level, it is also possible to display which test it is, when and by whom it was performed. In the context of the present prototype this was achieved by a manual entry in the corresponding test result XML file to show that it is possible and how it would look like. In a real implemented OQ Test App the tester could enter this via an appropriate user interface, if the

data could not be automatically taken over by the system (e.g. tester identification/authentication by single-sign-on).

2. The feature level displays the test results at the level of the scenarios that are intended to test the corresponding feature:

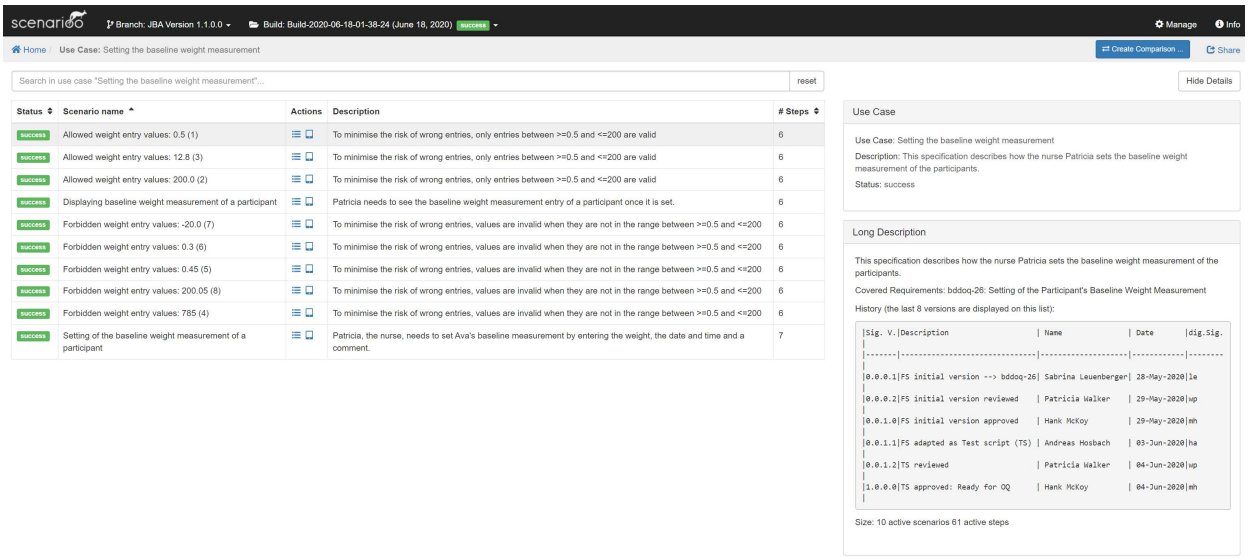


Figure 43: Overview of all scenarios of a feature

In addition, information from the specification brief section of the feature file is displayed automatically by the Cucumber-Scenarioo-plugin. This information allows the test reviewer to check the completeness of the tests performed by the OQ Test App. In this context it is important that the expected extent (size) of scenarios and steps is specified in the specification brief:

```
Feature: Setting the baseline weight measurement
  This specification describes how the nurse Patricia sets the baseline weight measurement of the participants.

  Covered Requirements:
  bddoq-26: Setting of the Participant's Baseline Weight Measurement

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-26	Sabrina Leuenberger	28-May-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	29-May-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	29-May-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	03-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	04-Jun-2020	wp
1.0.0.0	TS approved: Ready for OQ	Hank McKoy	04-Jun-2020	mh
  ```

  Size:
  10 active scenarios
  61 active steps
```

Figure 44: Indication of the testing extend to control the test coverage of the test runner

3. The scenario level shows an overview of the individual steps of a scenario with the corresponding screenshots:

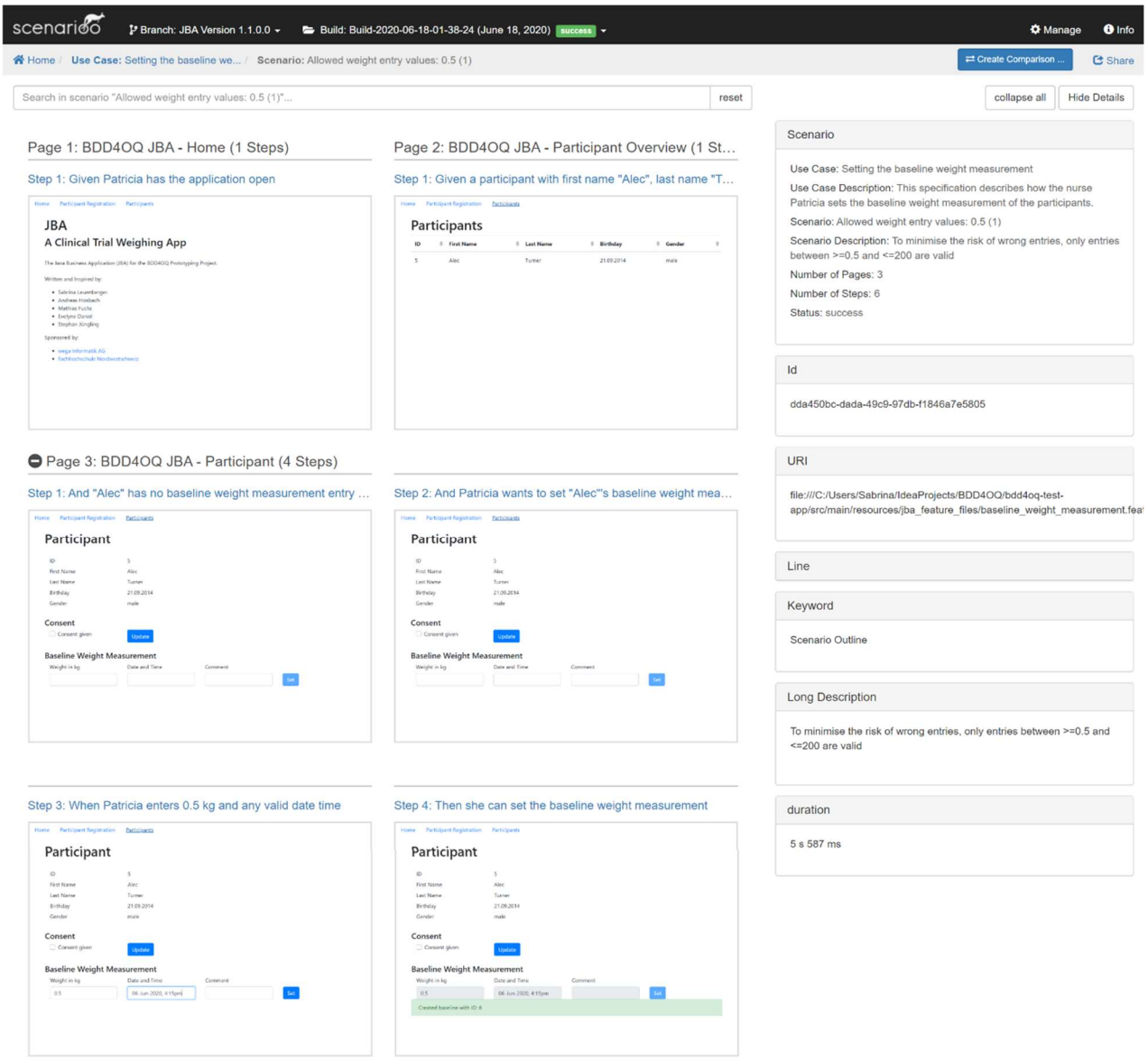


Figure 45: Step overview of one scenario

4. The step level displays the details of every single step:

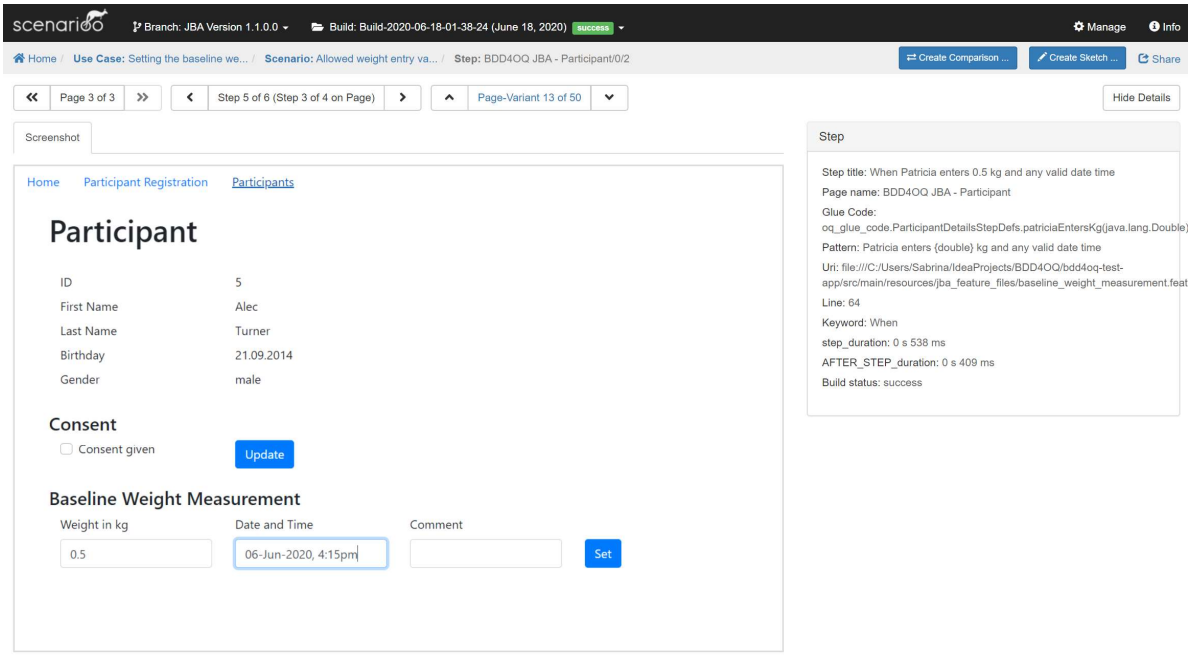


Figure 46: Detail view of a step

This view allows the reviewer to examine the test results in great detail: On one side the screenshot is displayed, on the other side in the block 'Step' information from the feature file, e.g. 'Step title' and data about the test execution, e.g. 'step_duration' and 'Build status' are displayed. Contradictions between the displayed data and what is shown in the screenshot indicate that there is an error. This must then be investigated and evaluated in the test report described below.

This hierarchical structure of the test visualisation with additional data from the feature file and from the actual test execution should allow the test reviewer to perform a targeted and an in-depth review of the test results.

6.7.2 Test Report

Based on the test review, the test reviewer writes a test report. For the prototype, this was developed as a form which must be filled out by the test reviewer (see Appendix XVII)

6.8 Implications for the Automated OQ when Adding New Functionalities

In the context of implementing a custom software it is quite conceivable that the need arises to adapt a functionality or to extend the software. During the creation of the prototype, the impact of such changes on the OQ documents was also examined. What was not part of the investigation, however, was the associated change management process, which would also have to be taken into account in a real project.

Two situations were investigated:

1. A change should be made before the OQ were performed.
2. After a first OQ has been performed (JBA version 1.0.0.0), the application was to be extended, resulting in a second OQ over JBA version 1.1.0.0.

6.8.1 Change Before the OQ

If it is decided during the development phase that a change should be made, this can be introduced via a new requirement. The requirement is specified and formulated as a scenario. This new scenario is then either documented on a new feature file or, if it is only an extension of an existing functionality, added to the corresponding feature file, as shown in Figure 47.

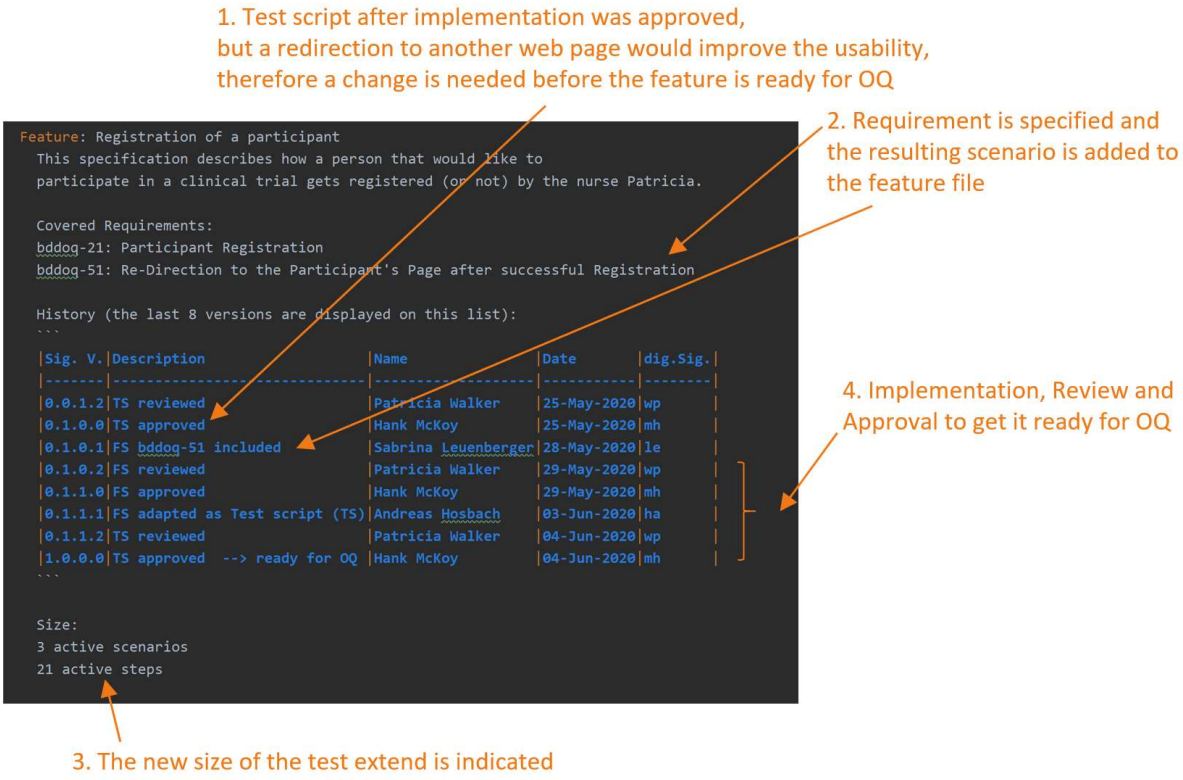


Figure 47: Adaptations in the specification brief due to the addition of a new requirement

6.8.2 Extension of JBA After the OQ

If new functionalities are added to the application after an OQ has been performed, this usually means that a new OQ must be prepared based on a risk assessment. This OQ necessarily runs over parts of the application that have already been through a former OQ and parts for which it is the first OQ. For the features that were already implemented, this re-executed OQ tests represent a regression testing (Guru99, 2020c).

Provided that the corresponding feature files with the associated glue code have not changed since the last OQ and have been found to be 'fit for intended use' within the scope of this former OQ, OQ automation can be relied upon. Therefore, a more in-depth review process is not required for these functionalities. It is sufficient to check whether all tests have been performed and whether the automation has found errors (partial test review: see Appendix XV, page 5 of the Test Specification). These would then be described and evaluated in the test report.

A more in-depth review is necessary for the functionalities that have been added, which in addition to the JBA OQ also aims to check the feature files and the glue code. In this sense, a review is necessary in this case, which checks every single step (full test review see Appendix XV, page 5 of the Test Specification).

In addition to the new functionalities, it may also be useful to subject certain key functionalities to a second full test review. This category could include functionalities that in one way or another interacts with the new functionality and therefore needs to be tested for integration, or functionalities that are of particular business criticality and therefore needs to be fully tested again.

In the prototype such an extension of the JBA was implemented and the audit was performed on this extended JBA (version 1.1.0.0). The concrete procedure for the OQ of this extended prototype was defined in the test specification with the two corresponding forms (Appendices XV-XVII).

6.9 OQ Execution

As described in chapter 3, the type of OQ execution (manual, automated) must be defined and planned. Within the scope of this OQ, two forms were developed in addition to a test specification, which were tailored to OQ test automation based on BDD as described above. This test specification with the two forms can be found in the Appendices XV-XVII.

7 Prototype Audit

The OQs on JBA version 1.1.0.0 were audited to assess their GAMP5 compliance. This audit was performed by wega CSV expert Evelyne Daniel. The focus was the OQ process with the following documents: Test Specification, JBA Test Execution: Results, Test Report and feature files. The displayed test results in Scenarioo were also assessed.

Based on that audit, Evelyne Daniel wrote an audit report that can be found in the Appendix XIV.

7.1 Results of the Audit

As the audit report shows, the prototype meets the GxP quality standards (Appendix XIV). At the same time, however, points that should be improved (minor findings) or recommendations that would lead to an amelioration (recommendations) were also recorded.

Four of the seven findings concern documentation. This documentation was already adapted accordingly during the audit and improved again in a subsequent review process with the help of Evelyne Daniel. This improved version of the ‘Test Specification’ with the two forms 'JBA Test Execution: Results' and 'Test Report' can be found in the Appendices XV-XVII. Examples of executed test results and test report were not attached but can be viewed in the GitHub Repository of the project (see ‘Reference to the Project Repository’ on the last page of this thesis).

Further three findings concern the actual prototype:

- The OQ Test App should contain a log of the activities that were conducted with the application (minor).
- The test results should be stored in the same location where Scenarioo can access them, so that the test results do not have to be moved manually (minor).
- Scenarioo provides an out-of-the-box functionality that allows automated comparison of different test runs. This functionality was not considered for the prototype but has the potential to improve the review process (recommendation).

7.2 Conclusions from the audit

On the basis of the audit report it can be stated that the developed prototype with the corresponding OQ documentation (the test specification including the two forms) is promising. The audit revealed points that need to be improved, but none of them would be showstopper. As a conclusion after the audit has been carried out, it can still be stated that nothing could be found that would contradict an OQ automation with the involvement of BDD in respect of GAMP5 compliance.

8 Learnings & Discussion

In order to answer the initial question of this project ‘BDD – A practicable Approach for CSV?’ the following hypothesis was investigated:

“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”.

For this purpose, four research questions in particular were dealt with:

- Do the artefacts out of the BDD-OQ process satisfy the GAMP5 OQ requirements?
- How can the executable requirement suite be adapted to the evolution of the application?
- Can automation tools like Cucumber/Gherkin, Selenium and Scenarioo be used together in validated environments?
- How could be dealt with new versions of the automation tools in terms of validation?

This was achieved in three incremental steps, which made it possible to examine the question with increasing depth and in a critical manner:

1. Theoretical consideration of the issue on the basis of GAMP5 and the BDD process: The result was a combined process based on the OQ GAMP5 process and integrating BDD elements that would allow automated OQ (see chapters 3 to 5, especially Figure 19).
2. Implementation of a prototype with respective OQ documentation based on the process developed in the first step: The creation of a prototype with automated OQ execution was possible, so that it could be shown that the theoretically developed process could also be implemented in practice (see chapter 6).
3. Evaluation of the prototype with the associated OQ documents with regard to GxP compliance as required for the regulated environment of the pharmaceutical industry (see chapter 7), that demonstrated, that such a OQ test system can indeed be implemented in a GxP compliant way.

To come back to the four research questions it can be stated that:

- The artefacts resulting from the BDD-OQ process, i.e. the feature files, the glue code, the test results in Scenarioo and the associated OQ documentation could be made GAMP5 compliant (see Appendix XIV).
- The executable specification suite can be adapted relatively easily to newly arising requirements, as shown in chapter 6.8. Additionally, without having elaborated on this, it can reasonably be assumed that these adaptations are easier to make than in a classical approach,

where the functional specification document needs to be adapted based on the new user requirements and the OQ test script must then be supplemented accordingly – and not to forget - the traceability that also needs to be ensured. Whereas in BDD, the user requirements can be incorporated directly into a feature file.

- According to a theoretical estimation, the test automation tool could be validated. This would make them suitable for use in the regulated environment of the pharmaceutical industry. However, an actual validation is still pending and would have to be carried out in a separate project. The approach on which such a project could build on can be found in chapter 6.3.
- The OQ Test App has hardly any logic of its own; it consists mainly of already existing components and libraries. These will normally be further developed and there could be a danger that the OQ Test App could start to make use of a different version than the one with which it was validated. Because of this, the prototype was designed as a Maven project, in which the version of the components and libraries to be included can be clearly defined via the POM file (see Appendix III). There is no risk of an automatic subsequent update because the OQ Test App was designed as an on-premise application that is deployed in a controlled environment (see chapter 6.4). The glue code and the feature files, which need a regular update, can be handled separately of the actual OQ Test App updates (see chapter 6.4).
- Scenarioo is an on-premise application, which allow to handle and manage all updates in controlled way by the regulated company (see chapter 6.4), therefore continuous updates in Scenarioo would not pose any problem to the regulated company.

Thus, the hypothesis defined at the beginning can be confirmed: BDD with its three practices can be used to execute a GAMP5 compliant OQ automation, as far as it has been tested and evaluated during the creation of the prototype.

In addition, the following supplementary insights were gained:

- It needs coding guidelines for the creation of the glue code as well as for the GUI of the Web application. For example, you can determine that
  - The tests are only controlled via the GUI, i.e. via Selenium, so that only tests are executed as they would be executed by a human. This also guarantees that the app is tested as it will be used by the users in the future (see chapter 6.6.1).
  - The GUI elements are tagged with a unique and descriptive ID to make their access via Selenium easier and more reliable (see chapter 6.6.1).
  - The design of the Web interface should not only be user-friendly, but should also help to simplify the review process, for example by making it easy to distinguish active from inactive buttons.

- It cannot be excluded that the feature files and/or the glue code has to be refactored, especially because test refactoring is an important principle of TDD (Nagy and Rose, 2018, p.15). It has to be taken into account that in the controlled environment of the highly regulated pharmaceutical companies, refactoring is always a risk and therefore it might imply extended revalidation in the worst case. Therefore, refactoring should be avoided whenever possible. If it has to be done, the refactored glue code / feature file should be run over the already tested and corresponding version of JBA. Meaning, it is only the refactoring itself in which the changes consists, so that only the refactoring needs to be checked in the subsequent review and approval process. This check could be done in a very similar way as for the automated OQs, as this already includes verifying activities for scenarios and for new or changed glue code.
- The review process could most likely significantly be improved or simplified by using the compare function of Scenarioo.

9 Outlook

This project did not only respond to questions, in fact even more new and further questions arose which could not be answered within the scope of this project. In the following these topics will therefore be raised and where possible first thoughts and approaches presented.

9.1 Added Value: OQs on several Web Browsers

OQ automation could become especially beneficial if the JBA is to run on different Web browsers and thus the same tests are to be performed several times in different browsers.

In the prototype, only Chrome was used as the browser for the JBA. The executing Web browser is defined by calling the corresponding Web driver as shown in Figure 48.

```
private TestContext() {
    System.setProperty("webdriver.chrome.driver", "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chromedriver.exe");
    webDriver = new ChromeDriver();
    webDriver.manage().window().setSize(new Dimension( width: 1024,  height: 768));
    webDriver.manage().timeouts().pageLoadTimeout( l: 5, TimeUnit.SECONDS);
    webDriver.manage().timeouts().implicitlyWait( l: 5, TimeUnit.SECONDS);

    RestAssured.baseURI = "http://localhost";
    RestAssured.port = 8098;
}
```

Figure 48: Call of the Chrome Web driver to perform the tests in Chrome.

It can be assumed that it should not be too difficult to extend the test context shown in Figure 48 in such a way that additional Web drivers would be called to perform these tests in parallel on several browsers. This could increase the flexibility with regard to the approved Web browsers for accessing the application.

9.2 Further Topics to Address

This project focused on the development of a Category 5 software and did not address a concrete process model. Therefore, it would be interesting to clarify how it would prove itself in slightly different situations, such as:

- Automated OQ Process with BDD for Category 4 Software
- BDD OQ Testing for Device Integration
- BDD OQ Testing in Agile Projects
- BDD OQ Testing and DevOps
- Impact on the automated OQ testing on time-to-market for a new software
- BDD Testing for PQ

9.3 Pharmaceutical Companies show Interest

During the implementation of this project it turned out that this project met high interest in the entourage of wega. Especially two pharmaceutical companies gave positive signals as well as wega employees working in projects of pharmaceutical companies. In personal conversations it became apparent that it would be worthwhile to continue to pursue the idea of BDD-Test Automation in the pharmaceutical community, for the verification of software according to GAMP5 and for other areas as well.

9.4 Automated OQ Testing and Artificial Intelligence

In the testing community there are efforts not only to automate testing more and more, but also to include an AI (Khan, 2020). It may well be worth investigating whether the newly developed OQ process could be simplified or made more efficient by including a test AI.

9.5 Towards a Digital Transformation of the Software Verification Process

In creating the OQ process, taking into account BDD practices and automation, the GAMP5 OQ process was based on the same principles as manual OQ execution. Only those adjustments were made that were minimally necessary to achieve GxP-compliant automation. This had the advantage that the process included steps that made it easy to have the glue code and feature files checked and approved by the QA. In consequence, this process still includes many manual steps. But nevertheless, it can be seen as a first step towards digital transformation of the software verification process.

With regard to a further digital transformation of the software verification process, it is conceivable that starting with the build process of the software, through its installation and the IQ, OQ and perhaps even PQ, all steps would become fully automated. This will undoubtedly only be possible, if the overall QA approach, of review and approval as described in GAMP5, will change. This will impact processes and even more importantly people who will need to adapt and this in a much more profound way than described in this project for the role of the tester[7].

However, there is still a long way to go in this respect, where many clarifications and investigations will be needed. Nevertheless, it would be exciting and promising to approach this vision step by step,

[7] There are some interesting articles dealing with this topic that can be found on following Web sites: https://dcatvci.org/6279-digital-transformation-in-2020-what-should-pharma-know - https://www2.deloitte.com/global/en/pages/life-sciences-and-healthcare/articles/gx-digital-transformation-in-life-sciences.html - https://www.pharma-iq.com/informatics/articles/digital-transformation-how-to-achieve-cultural-change.

for example by starting with the suggestions made before and always keeping an open mind for new ideas and options.

In any case, such a transformation would also have to be accompanied by a change in the prevailing culture of the pharmaceutical industry (Reilly, 2020), which would require a lot of sensitivity and a gradual approach.

10 Recommendation to the Attention of wega Informatik AG

In this project it was shown that OQ test automation based on BDD practices can be implemented in a GxP compliant manner. The interest shown in this topic from different sides and the fact that wega's customers signalled that they would be interested to participate, shows that it would be worthwhile to build on the experience gained and to pursue this topic further.

In this sense, the next step would be to investigate the real potential of the approach, such as improved quality, increased flexibility, increased efficiency, reduced 'time-to-market' or in the sense of this project - reduced 'time-to-production'.

In order to determine the effective added value, it would also be important to compare the expected benefit with the additional work required, such as the validation of the OQ Test App, the verification process of the glue code and generally the maintenance of the executable specification suite.

Since the prototype developed here is still too far away from reality, it was unfortunately not possible to answer such questions. Therefore, in this next step it would be important to start a project together with the industry, which could be carried out parallel to a real validation project, so that a concrete comparison in a practice-relevant situation would become possible.

References

AdoptOpenJDK. (n.d.). Prebuilt OpenJDK Binaries for Free! Retrieved 6 April 2020, from https://adoptopenjdk.net/index.html?variant=openjdk14&jvmVariant=hotspot

Agile Alliance. (2019, September 27). What are the Three Amigos in Agile? Retrieved 6 July 2020, from https://www.agilealliance.org/glossary/three-amigos

Ahmed, J. (2019, November 18). JUnit vs. TestNG: Choosing a Framework for Unit Testing. Retrieved 29 June 2020, from https://www.stickyminds.com/article/junit-vs-testng-choosing-framework-unit-testing

Anish. (2018, March 18). Create Cucumber Test Runner class. Retrieved 28 June 2020, from http://www.automationtestinghub.com/cucumber-test-runner-class-junit/

Atlassian. (n.d.-a). Confluence - Accomplish more together. Retrieved 7 July 2020, from https://www.atlassian.com/software/confluence

Atlassian. (n.d.-b). Jira | Issue & Project Tracking Software. Retrieved 7 July 2020, from https://www.atlassian.com/software/jira

Blaze Systems. (n.d.). Blaze IQ, OQ, PQ. Retrieved 19 April 2020, from https://www.blazesystems.com/blaze-iq-oq-pq.html

BootstrapVue. (n.d.). Components groups. Retrieved 28 June 2020, from https://bootstrap-vue.org/docs/components

Brown, S. (n.d.). The C4 model for visualising software architecture. Retrieved 6 April 2020, from https://c4model.com/

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

ChromeDriver Users. (2015, June 11). ChromeDriver Architecture. Retrieved 27 June 2020, from https://groups.google.com/forum/#!msg/chromedriver-users/xVMy5OGLcl8/2JljtZ1FAAAJ

Chromium. (n.d.). ChromeDriver - WebDriver for Chrome. Retrieved 6 April 2020, from https://chromedriver.chromium.org/

Coveros. (2014, April 14). Exploring Glue Code with Cucumber-JVM. Retrieved 28 June 2020, from https://www.coveros.com/exploring-glue-code-with-cucumber-jvm/

Cucumber. (n.d.-a). Cucumber Open - Get Started with BDD Today. Retrieved 29 June 2020, from https://cucumber.io/tools/cucumber-open/

Cucumber. (n.d.-b). Reporting - Cucumber Documentation. Retrieved 7 July 2020, from https://cucumber.io/docs/cucumber/reporting/

draw.io. (n.d.). draw.io - we love diagrams. Retrieved 7 July 2020, from https://drawio-app.com/

Esch, P. M., Donzé, G., Eschbach, B., Hassler, S., Hutter, L., Saxer, H. P., … Zühlke, R. (2007). Good Laboratory Practice (GLP) - guidelines for the validation of computerised systems. The Quality Assurance Journal, 11(3–4), 208–220. https://doi.org/10.1002/qaj.426

FDA. (2018, January 4). Informed Consent for Clinical Trials. Retrieved 10 July 2020, from https://www.fda.gov/patients/clinical-trials-what-patients-need-know/informed-consent-clinical-trials

Flenner, Y. (2020, February 26). What's So Great About JUnit 5? Retrieved 7 July 2020, from https://blog.testproject.io/2019/02/26/junit-5/

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

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

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

Guru99. (2020c, June 15). What is Regression Testing? Definition, Test Cases (Example). Retrieved 13 July 2020, from https://www.guru99.com/regression-testing.html

H2. (n.d.). H2 Database Engine. Retrieved 28 June 2020, from https://www.h2database.com/html/main.html

Hacker Girl. (2016, April 23). How to click on hidden element in Selenium WebDriver. Retrieved 17 July 2020, from https://medium.com/@hacker_girl/how-to-click-on-hidden-element-in-selenium-webdriver-873773dc333c

Hellesøy, A. (2015, March 24). Are you doing BDD? Or are you just using Cucumber? Retrieved 9 July 2020, from https://cucumber.io/blog/bdd/single-source-of-truth/

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

Hosbach, A. (2020, July 20). Cucumber-Scenarioo-plugin. Retrieved 4 May 2020, from https://github.com/andreashosbach/cucumber-reporter

ISPE. (2008). GAMP 5 - A Risk-Based Approach to Compliant GxP Computerized Systems. North Bethesda, United States of America: ISPE.

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

javaTpoint. (n.d.). Spring Boot JPA - javatpoint. Retrieved 18 July 2020, from https://www.javatpoint.com/spring-boot-jpa

Jet Brains. (2019, July 24). IntelliJ IDEA: The Java IDE for Professional Developers. Retrieved 6 April 2020, from https://www.jetbrains.com/idea/

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

Kazeeva, A. (2018, March 14). 10 Best Open Source Test Automation Frameworks for Every Purpose. Retrieved 7 July 2020, from https://dzone.com/articles/10-best-open-source-test-automation-frameworks-for

Ketterlin Fisher, C. (2019, February 28). 3 open source behavior-driven development tools. Retrieved 7 July 2020, from https://opensource.com/article/19/2/behavior-driven-development-tools

Khan, A. (2020, June 2). Artificial Intelligence is changing Test Automation and Micro Focus is leading the way. Retrieved 20 June 2020, from https://community.microfocus.com/t5/Application-Delivery-Management/Artificial-Intelligence-is-changing-Test-Automation-and-Micro/bap/2802102

Microsoft. (n.d.). Microsoft Teams. Retrieved 7 July 2020, from https://www.microsoft.com/en/microsoft-365/microsoft-teams/group-chat-software

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.

North, D. (2006, June 5). Introducing BDD. Retrieved 6 July 2020, from https://dannorth.net/introducing-bdd/

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

Plagiannos, C. (2015, November 30). What is GAMP®5 and how do I use it effectively? Retrieved 20 July 2020, from https://blog.montrium.com/experts/what-is-gamp5-and-how-do-i-use-it-effectively

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

Reilly, D. (2020, June 1). It's all in the mind(set): digital transformation in pharma. Retrieved 18 July 2020, from https://pharmafield.co.uk/in_depth/its-all-in-the-mindset-digital-transformation-in-pharma/

Rose, S. (2015, October 26). Introduction to BDD with Cucumber for Java. Retrieved 7 July 2020, from https://www.slideshare.net/sebrose/introduction-to-bdd-with-cucumber-for-java

Rose, S., Wynne, M., & Hellesøy, A. (2015). The Cucumber for Java Book. Raleigh, United States of America: Pragmatic Bookshelf.

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

Scenarioo. (n.d.-b). Setup of Scenarioo Viewer Web App. Retrieved 29 June 2020, from http://scenarioo.org/docs/master/tutorial/Scenarioo-Viewer-Web-Application-Setup.html

Selenium. (n.d.). Selenium automates browsers. That's it! Retrieved 27 March 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 27 March 2020, from https://cucumber.io/

SoftwareTestingHelp. (2020, June 30). 8 Best Behavior Driven Development (BDD) Tools and Testing Frameworks. Retrieved 7 July 2020, from https://www.softwaretestinghelp.com/behavior-driven-development-bdd-tools/

Spring. (n.d.). Spring Boot. Retrieved 28 June 2020, from https://spring.io/projects/spring-boot

stackoverflow. (2018, September 20). Difference between OpenJDK and Adoptium/AdoptOpenJDK. Retrieved 6 April 2020, from https://stackoverflow.com/questions/52431764/

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

Tutorialspoint. (n.d.-a). Cucumber - Gherkins - Tutorialspoint. Retrieved 28 June 2020, from https://www.tutorialspoint.com/cucumber/cucumber_gherkins.htm

Tutorialspoint. (n.d.-b). Cucumber - JUnit Runner - Tutorialspoint. Retrieved 28 June 2020, from https://www.tutorialspoint.com/cucumber/cucumber_junit_runner.htm

Tutorialspoint. (n.d.-c). Selenium - Webdriver - Tutorialspoint. Retrieved 28 June 2020, from https://www.tutorialspoint.com/selenium/selenium_webdriver.htm

Tyson, M. (2019, April 2). What is JPA? Introduction to the Java Persistence API. Retrieved 18 July 2020, from https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

Unadkat, J., & Krishnakumar, P. (2019, July 8). Selenium Webdriver Tutorial with Examples. Retrieved 26 June 2020, from https://www.browserstack.com/guide/selenium-webdriver-tutorial

VMware. (2020, January 1). Spring makes Java... Retrieved 27 March 2020, from https://spring.io/

Vue.js. (n.d.). Vue.js. Retrieved 28 June 2020, from https://vuejs.org/

Wikipedia. (n.d.). OpenJDK. Retrieved 6 April 2020, from https://en.wikipedia.org/wiki/OpenJDK

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

Wikipedia. (2020, May 11). Cucumber (software). Retrieved 29 June 2020, from https://en.wikipedia.org/w/index.php?title=Cucumber_(software)&oldid=956132812

Wyn, S. (2018). GAMP 5: Ten years on. Pharmaceutical Engineering, 38(3), 18–19. Retrieved from https://ispe.org/pharmaceutical-engineering/may-june-2018

Wynne, M. (n.d.). Introduction to BDD Example Mapping. Retrieved 6 July 2020, from https://cucumber.io/blog/bdd/example-mapping-introduction/

List of Figures

List of Abbreviations

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
BDD	Behaviour Driven Development:
BPMN	Business Process Model and Notation
CSV	Computerised System Validation
DB	Database
DOM	Document Object Model
FDA	U.S. Food and Drug Administration
FRA	Functional Risk Assessment
GAMP	Good Automated Manufacturing Practice
GAMP5	GAMP Guide “A Risk-Based Approach to Compliant GxP Computerized Systems”
GxP	Good x Practice - x acts as a placeholder for the different practices. For example: AM for Automated Manufacturing C for Clinical L for Laboratory M for Manufacturing
IQ	Installation Qualification
ISPE	International Society for Pharmaceutical Engineering: World’s largest not-for-profit association serving its members by leading scientific, technical and regulatory advancement throughout the entire pharmaceutical lifecycle.
JBA	Java Business App
JDK	Java Development Kit
JPA	Java Persistence API
JVM	Java Virtual Machine

PNG	Portable Network Graphics
OQ	Operational Qualification
POM	Project Object Model
PQ	Performance Qualification
QA	Quality Assurance
Rest	Representational state transfer
SME	Subject Matter Expert
SOP	Standard Operating Procedure
StepDefs	Step definitions
TDD	Test Driven Development
XML	Extensible Markup Language

Appendix I: Helpful Sites Used for the Prototype Implementation

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/

Appendix II: Version control files JBA (POM/package.json)

JBA POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://ma-
ven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.github.sableu</groupId>
    <artifactId>bdd4oq-jba</artifactId>
    <packaging>pom</packaging>
    <version>1.1.0</version>
    <modules>
        <module>backend</module>
        <module>frontend</module>
    </modules>
</project>
```

JBA Frontend POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://ma-
ven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>bdd4oq-jba</artifactId>
        <groupId>com.github.sableu</groupId>
        <version>1.1.0</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>frontend</artifactId>
    <packaging>pom</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <frontend-maven-plugin.version>1.9.1</frontend-maven-plugin.version>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>com.github.eirslett</groupId>
                <artifactId>frontend-maven-plugin</artifactId>
                <version>${frontend-maven-plugin.version}</version>
                <executions>
                    <!-- Install our node and npm version to run npm/node scripts-->
                    <execution>
                        <id>install node and npm</id>
                        <goals>
                            <goal>install-node-and-npm</goal>
                        </goals>
                        <configuration>
                            <nodeVersion>v12.12.0</nodeVersion>
                        </configuration>
                    </execution>
                    <!-- Install all project dependencies -->
```

```
                    <execution>
                        <id>npm install</id>
                        <goals>
                            <goal>npm</goal>
                        </goals>
                        <!-- optional: default phase is "generate-resources" -->
                        <phase>generate-resources</phase>
                        <!-- Optional configuration which provides for running any npm
command -->
                        <configuration>
                            <arguments>install</arguments>
                        </configuration>
                    </execution>
                    <!-- Build and minify static files -->
                    <execution>
                        <id>npm run build</id>
                        <goals>
                            <goal>npm</goal>
                        </goals>
                        <configuration>
                            <arguments>run build</arguments>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Package.json

```
{
  "name": "frontend",
  "version": "1.1.0",
  "description": "Vue.js frontend",
  "private": true,
  "scripts": {
    "clean": "rm -rf node_modules target",
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  },
  "dependencies": {
    "axios": "0.19.2",
    "bootstrap": "4.4.1",
    "bootstrap-vue": "2.11.0",
    "jquery": "3.5.0",
    "core-js": "3.6.5",
    "vue": "2.6.11",
    "vue-router": "3.1.6",
    "vuex": "3.1.3"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "4.3.1",
    "@vue/cli-plugin-router": "4.3.1",
    "@vue/cli-plugin-vuex": "4.3.1",
    "@vue/cli-service": "4.3.1",
    "node-sass": "4.13.1",
    "sass-loader": "8.0.2",
    "vue-template-compiler": "2.6.11"
  }
}
```

JBA Backend POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://ma-
ven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>bdd4oq-jba</artifactId>
        <groupId>com.github.sableu</groupId>
        <version>1.1.0</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>backend</artifactId>
    <packaging>jar</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
        <spring-boot.version>2.2.7.RELEASE</spring-boot.version>
        <h2.version>1.4.200</h2.version>
        <rest-assured.version>4.3.0</rest-assured.version>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven-resources-plugin.version>3.1.0</maven-resources-plugin.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <version>${spring-boot.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <optional>true</optional>
            <version>${spring-boot.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
            <version>${spring-boot.version}</version>
        </dependency>

        <!-- In-Memory database used for local development & testing -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>${h2.version}</version>
        </dependency>

        <dependency>
            <groupId>jakarta.xml.bind</groupId>
            <artifactId>jakarta.xml.bind-api</artifactId>
            <version>2.3.2</version>
        </dependency>

        <!-- Runtime, com.sun.xml.bind module -->
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
            <version>2.3.2</version>
```

```
        </dependency>

        <!-- Testing -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <version>${spring-boot.version}</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>io.rest-assured</groupId>
            <artifactId>rest-assured</artifactId>
            <version>${rest-assured.version}</version>
            <scope>test</scope>
        </dependency>


    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>${spring-boot.version}</version>
            </plugin>
            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>${maven-resources-plugin.version}</version>
                <executions>
                    <execution>
                        <id>copy Vue.js frontend content</id>
                        <phase>generate-resources</phase>
                        <goals>
                            <goal>copy-resources</goal>
                        </goals>
                        <configuration>
                            <outputDirectory>src/main/resources/public</outputDirectory>
                            <overwrite>true</overwrite>
                            <resources>
                                <resource>
                                    <directory>${project.parent.basedir}/frontend/tar-
get/dist</directory>
                                    <includes>
                                        <include>static/</include>
                                        <include>index.html</include>
                                    </includes>
                                </resource>
                            </resources>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Appendix III: Version control file OQ Test App (POM)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://ma-
ven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.github.sableu</groupId>
    <artifactId>bdd4oq-test-app</artifactId>
    <version>0.0.1</version>
    <packaging>jar</packaging>

    <name>oq-test.app</name>
    <description>Test automation tool to perform OQs on a web application</description>
    <url>https://github.com/sableu/BDD4OQ</url>

    <licenses>
        <license>
            <name>The Apache License, Version 2.0</name>
            <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
        </license>
    </licenses>

    <developers>
        <developer>
            <name>Sabrina Leuenberger</name>
            <email>saleuenberger@gmx.ch</email>
            <organizationUrl>https://github.com/sableu</organizationUrl>
        </developer>
    </developers>

    <scm>
        <connection>scm:git:git://github.com/sableu/BDD4OQ</connection>
        <developerConnection>scm:git:ssh://github.com/sableu/BDD4OQ</developerConnection>
        <url>https://github.com/sableu/BDD4OQ</url>
    </scm>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <cucumber-version>5.1.3</cucumber-version>
        <gherkin-version>9.0.0</gherkin-version>
        <rest-assured.version>4.3.0</rest-assured.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.6.0</version>
        </dependency>

        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-launcher</artifactId>
            <version>1.6.2</version>
        </dependency>

        <dependency>
```

```
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
            <version>5.6.2</version>
        </dependency>

        <dependency>
            <groupId>io.rest-assured</groupId>
            <artifactId>rest-assured</artifactId>
            <version>${rest-assured.version}</version>
        </dependency>

        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>${cucumber-version}</version>
        </dependency>

        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-core</artifactId>
            <version>${cucumber-version}</version>
        </dependency>

        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-junit</artifactId>
            <version>${cucumber-version}</version>
        </dependency>

        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>gherkin</artifactId>
            <version>${gherkin-version}</version>
        </dependency>

        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
        </dependency>

        <dependency>
            <groupId>com.github.andreashosbach</groupId>
            <artifactId>cucumber-scenarioo-plugin</artifactId>
            <version>0.5.0</version>
        </dependency>

    </dependencies>

</project>
```

Appendix IV: Feature File – Participant Overview

```
#language: en

Feature: Participant's overview
  An overview of all registered participants is displayed, so that starting from this
overview, the baseline weight measurement can be set for the participants where it was
not done so far

  Covered Requirements:
  bddoq-50: Overview (List) of all Registered Participants

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-50	Sabrina Leuenberger	23-May-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	24-May-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	25-May-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	03-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	04-Jun-2020	wp
1.0.0.0	TS approved --> ready for OQ	Hank McKoy	04-Jun-2020	mh
  ```

  Size:
  1 active scenario
  3 active step


  Background:
    Given Patricia has the application open

  Scenario: Overview of all registered participants

    Given participants with first name, last name, birthday, gender are registered
    | "Natasha" | "Romanoff" | "1st of January 1984" | "female" |
    | "Scott"   | "Lang"     | "1st of March 1997"   | "male"   |
    | "Bruce"   | "Banner"   | "1st of May 1962"     | "male"   |
    | "Betty"   | "Ross"     | "1st of May 1962"     | "female" |
    Then the participants should be found in the overview
```

Appendix V: Feature File – Participant Registration

```
#language: en

Feature: Registration of a participant
  This specification describes how a person that would like to
  participate in a clinical trial gets registered (or not) by the nurse Patricia.

  Covered Requirements:
  bddoq-21: Participant Registration
  bddoq-51: Re-Direction to the Participant's Page after successful Registration

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.1.2	TS reviewed	Patricia Walker	25-May-2020	wp
0.1.0.0	TS approved	Hank McKoy	25-May-2020	mh
0.1.0.1	FS bddoq-51 included	Sabrina Leuenberger	28-May-2020	le
0.1.0.2	FS reviewed	Patricia Walker	29-May-2020	wp
0.1.1.0	FS approved	Hank McKoy	29-May-2020	mh
0.1.1.1	FS adapted as Test script (TS)	Andreas Hosbach	03-Jun-2020	ha
0.1.1.2	TS reviewed	Patricia Walker	04-Jun-2020	wp
1.0.0.0	TS approved --> ready for OQ	Hank McKoy	04-Jun-2020	mh
  ```

  Size:
  3 active scenarios
  21 active steps


  Background:
    Given Patricia has the application open


  Scenario: Registration of an unknown participant

  To register a new participant who has no entry in the DB yet, the first name, family
name, gender and birth date needs to be entered in order to successfully register the new
participant.
  This represents the simplest happy path.

    Given a participant with first name "Peter", last name "Parker", birthday
"10.08.2001" and is "male"
    And "Peter" is not registered yet
    And Patricia wants to register "Peter"
    When Patricia enters "Peter"'s data
    And registers "Peter"'s data
    Then "Peter"'s details should be displayed


  Scenario Outline: Registration of unknown participants with complicated names <descrip-
tion>

  Non standard characters, that might be used in some names, have to be accepted by the
system

    Given a participant with first name <first_name>, last name <last_name>, birthday
<birthday> and is <gender>
    And <first_name> is not registered yet
    And Patricia wants to register <first_name>
    When Patricia enters <first_name>'s data
    And registers <first_name>'s data
```

```
    Then <first_name>'s details should be displayed
    Examples:
      | description    | first_name      | last_name         | birthday                |
gender   |
      | "specialities" | "Hans-Peter J." | "Rudolf von Rohr" | "16th of May 1951"      |
"male"   |
      | "french"       | "Céline"        | "d'Artagnan"      | "18th of November 1982" |
"female" |


  @Ignore
  Scenario Outline: Denied registration of an unknown participant

  First name, family name, gender and birth date are mandatory fields and needs to be en-
tered in order to finalise the registration.
  The mandatory fields are defined in a way to mitigate the risk of a mix-up of two par-
ticipants.

    Given Alex is not registered yet
    And the registration form is displayed
    And the First Name field of the form has value <first_name>
    And the Last Name field of the form has value <last_name>
    And the birthday field of the form has value <birthday>
    And the Gender field of the form has value <gender>
    When Patricia confirms the registration
    Then The system should stay on the registration page
    And should display the message
          """
          Please fill in all mandatory fields!
          """
    And Alex should not be registered

    Examples:
      | first_name | last_name | birthday     | gender |
      | "Alex"     | "Turner"  | "21.09.1946" | ""     |
      | "Alex"     | "Turner"  | ""           | "male" |
      | "Alex"     | ""        | "21.09.1946" | "male" |
      | ""         | "Turner"  | "21.09.1946" | "male" |
      | "Alex"     | "Turner"  | ""           | ""     |
      | ""         | "Turner"  | ""           | ""     |


  @Ignore
  Scenario: Registration with denial of participation

  A person would like to participate in the clinical trial, but he/she is not allowed to.
He/she can still be registered in view of future clinical trials, where the person could
participate.

    Given Alicia is not registered yet
    And Alicia is not allowed to participate yet
    And Patricia wants to register her for a future clinical trial
    When Patricia enters the following dates and confirms
      | field                | value                                                                |
      | "First Name"         | "Alicia"                                                             |
      | "Last Name"          | "Bianchi"                                                            |
      | "Gender"             | "female"                                                             |
      | "Birthday"           | "9.2.1992"                                                           |
      | "Participation denied | "yes"                                                               |
      | "Comment"            | "Patricia is pregnant. She will be able to participate in
2022: alicia@foo.com" |
    Then Alicia should be found in the system with all the information entered.


  @Ignore
  Scenario: Update registration of known participant
```

```
  A person is already registered, but one or several of the entries needs to be updated.

    Given Mia is already registered as Mia Parker
    And Patricia wants to update her last name and add a comment
    When Patricia enters "Miller" in the respective field
    And adds into the comment field "married the 12th of June 2019"
    Then the new set of data should be found in the system


  @Ignore
  Scenario Outline: Denied registration of known participant

  A person is already registered, then it should not be possible to register him/her a
second time.
  To mitigate the risk of mixing up participants, the combination of the first name, last
name and birthday must be unique and case insensitive.

    Given Scott is already registered with following data: "Scott", "Lang", "1st of March
1997", "male"
    And Patricia wants to register Scott
    When Patricia enters <first_name>, <last_name>, <birthday>" and <gender> in the re-
spective fields
    Then The system should display the message
          """
          Entry already exists!
          """
    And the existing entry should show up.
    Examples:
      | first_name | last_name | birthday            | gender   |
      | "Scott"    | "Lang"    | "1st of March 1997" | "male"   |
      | "Scott"    | "Lang"    | "1st of March 1997" | "female" |
      | "Scott"    | "lang"    | "1st of March 1997" | "male"   |
      | "scott"    | "lang"    | "1st of March 1997" | "male"   |
      | "scott"    | "Lang"    | "1st of March 1997" | "male"   |


  @Ignore
  Scenario Outline: Accepted registration of a similar participant

  A person is already registered, then it should not be possible to register him/her a
second time.

    Given Scott is already registered with following data: "Scott", "Lang", "1st of March
1997", "male"
    And Patricia wants to register a similar participant
    When Patricia enters <first_name>, <last_name>, <birthday>" and <gender> in the re-
spective fields
    And wants to register them
    Then the similar participant should be found in the system
    Examples:
      | first_name | last_name | birthday             | gender |
      | "Scott"    | "Lang"    | "31st of March 1990" | "male" |
      | "Scotty"   | "Lang"    | "1st of March 1997"  | "male" |
      | "Scott"    | "Lango"   | "1st of March 1997"  | "male" |
```

Appendix VI: Feature File – Baseline Weight Measurement

```
#language: en

Feature: Setting the baseline weight measurement
  This specification describes how the nurse Patricia sets the baseline weight measure-
ment of the participants.

  Covered Requirements:
  bddoq-26: Setting of the Participant's Baseline Weight Measurement

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-26	Sabrina Leuenberger	28-May-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	29-May-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	29-May-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	03-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	04-Jun-2020	wp
1.0.0.0	TS approved: Ready for OQ	Hank McKoy	04-Jun-2020	mh
  ```

  Size:
  10 active scenarios
  61 active steps


  Background:
    Given Patricia has the application open


  Scenario: Setting of the baseline weight measurement of a participant
  Patricia, the nurse, needs to set Ava's baseline measurement by entering the weight,
the date and time and a comment.
  This represents the simplest happy path.

    Given a participant with first name "Ava", last name "Johnson", birthday
"01.01.1989", gender "female" is registered
    And "Ava" has no baseline weight measurement entry yet
    And Patricia wants to set "Ava"'s baseline weight measurement
    When Patricia enters 68.5 kg in the weight field, "15.5.20, 8:15am" in the time field
and "measurement done right after breakfast" in the comment field
    And she saves these entries
    Then "Ava"'s baseline weight entry should be found in the system

  Scenario: Displaying baseline weight measurement of a participant
  Patricia needs to see the baseline weight measurement entry of a participant once it is
set.

    Given a participant with first name "Ava", last name "Johnson", birthday
"01.01.1989", gender "female" is registered
    And "Ava"'s baseline weight measurement is set
    And Patricia is on the participants overview page
    When Patricia opens "Ava"'s detail page
    Then "Ava"'s baseline weight entry should be displayed on that page

  @Ignore
  Scenario: Denial of setting the baseline weight measurement when it already exists.

    Given Ava with first name "Ava", last name "Johnson", birthday "01.01.1989", gender
"female" is registered
    And Ava has already a baseline measurement
```

```
    When Patricia wants to register Ava's baseline weight measurement
    Then she should not be able to register a new baseline measurement

  Scenario Outline: Allowed weight entry values: <weight>
  To minimise the risk of wrong entries, only entries between >=0.5 and <=200 are valid

    Given a participant with first name <first_name>, last name <last_name>, birthday
"21.09.2014", gender "male" is registered
    And <first_name> has no baseline weight measurement entry yet
    And Patricia wants to set <first_name>'s baseline weight measurement
    When Patricia enters <weight> kg and any valid date time
    Then she can set the baseline weight measurement
    Examples:
      | first_name | last_name | weight |
      | "Alec"     | "Turner"  | 0.5    |
      | "Elec"     | "Turner"  | 200.0  |
      | "Ilex"     | "Turner"  | 12.8   |

  Scenario Outline: Forbidden weight entry values: <weight>
  To minimise the risk of wrong entries, values are invalid when they are not in the
range between >=0.5 and <=200

    Given a participant with first name "Eric", last name "Turner", birthday
"21.09.2014", gender "male" is registered
    And "Eric" has no baseline weight measurement entry yet
    And Patricia wants to set "Eric"'s baseline weight measurement
    When Patricia enters <weight> kg and any valid date time
    Then she cannot set the baseline weight measurement
    Examples:
      | weight |
      | 785    |
      | 0.45   |
      | 0.3    |
      | -20.0  |
      | 200.05 |
```

Appendix VII: Feature File – Consent Management

```
#language: en

Feature: Consent management
  This specification describes how the nurse Patricia manages the participant's consent.
  The participant's consent is an important aspect in order to achieve compliance with
HFV Art. 31 (https://www.eknz.ch/gesuchseinreichung/)

  Covered Requirements:
  bddoq-22: Consent registration

  History (the last 8 versions are displayed on this list):
  ```
Sig. V.	Description	Name	Date	dig.Sig.
0.0.0.1	FS initial version --> bddoq-22	Sabrina Leuenberger	11-Jun-2020	le
0.0.0.2	FS initial version reviewed	Patricia Walker	11-Jun-2020	wp
0.0.1.0	FS initial version approved	Hank McKoy	11-Jun-2020	mh
0.0.1.1	FS adapted as Test script (TS)	Andreas Hosbach	15-Jun-2020	ha
0.0.1.2	TS reviewed	Patricia Walker	15-Jun-2020	wp
1.0.0.0	TS approved --> ready for OQ	Hank McKoy	15-Jun-2020	mh
  ```

  Size:
  1 active scenarios
  6 active steps


  Background:
    Given Patricia has the application open

  Scenario: Consent registration
  Patricia, the nurse, needs to know, if a participant has given her consent for the
weight measurments and the subsequent data handling. Therefore she has to add that infor-
mation, after the consent was given.
  This represents the simplest happy path.

    Given a participant with first name "Wanda", last name "Maximoff", birthday
"12.12.1989", gender "female" is registered
    And "Wanda" did not give her consent so far
    When Patricia registers that "Wanda" gave her consent
        And she displays "Wanda"'s details
        Then Patricia should see on the participant detail page that the consent was
given.
```

Appendix VIII: Glue Code – Test Context

```
package oq_glue_code;

import io.restassured.RestAssured;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class TestContext {
    private static TestContext instance;

    public static TestContext getInstance() {
        if (instance == null) {
            instance = new TestContext();
        }
        return instance;
    }

    public static void cleanup() {
        if (instance != null) {
            instance.webDriver.close();
            instance.webDriver.quit();
            instance = null;
        }
    }

    public static WebDriver webDriver(){
        return getInstance().getWebDriver();
    }
    private List<Participant> participants =  new ArrayList<>();

    public static Participant participant(String firstName){
        return getInstance().getParticipant(firstName);
    }

    public static List<Participant> participants(){
        return getInstance().getParticipants();
    }

    public static void addParticipant(String firsName){
        Participant participant = new Participant();
        participant.firstName = firsName;
        getInstance().addParticipant(participant);
    }

    public static  void clearParticipants(){
        getInstance().getParticipants().clear();
    }
    private WebDriver webDriver;

    private TestContext() {
        System.setProperty("webdriver.chrome.driver", "C:\\Program Files
(x86)\\Google\\Chrome\\Application\\chromedriver.exe");
        webDriver = new ChromeDriver();
        webDriver.manage().window().setSize(new Dimension(1024, 768));
        webDriver.manage().timeouts().pageLoadTimeout(5, TimeUnit.SECONDS);
        webDriver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
```

```
        RestAssured.baseURI = "http://localhost";
        RestAssured.port = 8098;
    }

    private WebDriver getWebDriver() {
        return webDriver;
    }

    private Participant getParticipant(String firstName){
        return participants.stream().filter(p -> firstName.equals(p.firstName)).find-
First().get();
    }

    private List<Participant> getParticipants(){
        return participants;
    }

    private void addParticipant(Participant participant){
        this.participants.add(participant);
    }
}
```

Appendix IX: Glue Code – General StepDefs

```
package oq_glue_code;

import com.github.andreashosbach.cucumber_scenarioo_plugin.model.Screenshot;
import io.cucumber.java.After;
import io.cucumber.java.AfterStep;
import io.cucumber.java.Before;
import io.cucumber.java.en.Given;
import io.restassured.response.Response;
import io.restassured.response.ValidatableResponse;
import io.restassured.specification.RequestSender;
import org.apache.http.HttpStatus;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;

import static io.restassured.RestAssured.when;
import static oq_glue_code.TestContext.*;


public class GeneralStepDefs {
    @Before
    public void setup() {
        webDriver().navigate().to("about:blank");
    }

    @After
    public void cleanupData() {
        for(Participant participant : participants()){
            if (participant.id != null) {
                RequestSender sender = when();
                Response response = sender.delete("/api/participant/" + participant.id);
                ValidatableResponse vResponse = response.then();
                vResponse.statusCode(HttpStatus.SC_NO_CONTENT);
            }
        }
        clearParticipants();
    }

    @Given("Patricia has the application open")
    public void hasTheApplicationOpen() {
        webDriver().navigate().to("http://localhost:8098/");
    }

    @AfterStep
    public void afterStep() {
        TakesScreenshot scrShot = ((TakesScreenshot) (webDriver()));
        String title = webDriver().getTitle();
        Screenshot.save(title.isEmpty() ? "Unavailable" : title, scrShot.getScreen-
shotAs(OutputType.BYTES));
    }
}
```

Appendix X: Glue Code – Participant Overview StepDefs

```
package oq_glue_code;

import io.cucumber.datatable.DataTable;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import okhttp3.MultipartBody;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import static oq_glue_code.TestContext.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.MatcherAssert.assertThat;


public class ParticipantOverviewStepDefs {

    @Given("a participant with first name {string}, last name {string}, birthday {string}
and is {string}")
    public void aParticipantWithFirstNameLastNameBirthdayAndIs(String firstName, String
lastName, String birthday, String gender) {
        addParticipant(firstName);
        participant(firstName).lastName = lastName;
        participant(firstName).birthday = birthday;
        participant(firstName).gender = gender;
    }

    @Given("a participant with first name {string}, last name {string}, birthday
{string}, gender {string} is registered")
    public void aParticipantWithFirstNameLastNameBirthdayGenderIsRegistered(String first-
Name, String lastName, String birthday, String gender) {
        aParticipantWithFirstNameLastNameBirthdayAndIs(firstName, lastName, birthday,
gender);
        ParticipantRegistrationStepDefs.patriciaWantsToRegister(firstName);
        ParticipantRegistrationStepDefs.patriciaEntersData(firstName);
        ParticipantRegistrationStepDefs.registersThem(firstName);
        shouldBeFoundInTheOverview(firstName);
    }

    @Given("participants with first name, last name, birthday, gender are registered")
    public void participants_with_first_name_last_name_birthday_gender_are_regis-
tered(DataTable dataTable) {
        for (List<String> list : dataTable.asLists()) {
            aParticipantWithFirstNameLastNameBirthdayGenderIsRegistered(list.get(0),
list.get(1), list.get(2), list.get(3));
        }
    }

    @And("{string} is not registered yet")
    public void isNotRegisteredYet(String firstName) {
        assertThat(participantOverviewContainsParticipant(firstName), is(false));
    }

    @Then("{string} should be found in the overview")
    public void shouldBeFoundInTheOverview(String firstName) {
        assertThat(participantOverviewContainsParticipant(firstName), is(true));
```

```
    }

    private boolean participantOverviewContainsParticipant(String firstName) {
        webDriver().findElement(By.linkText("Participants")).click();
        List<WebElement> row = webDriver().findElements(By.xpath("//*[@id='partici-
pantTable']/tbody/tr/td[1]"));
        int numParticipants = row.size();
        WebElement participantTable = webDriver().findElement(By.id("participantTable"));
        for (int i = 1; i <= numParticipants; i++) {
            WebElement tableCellFirstName = participantTable.findEle-
ment(By.xpath("//*[@id='participantTable']/tbody/tr[" + i + "]/td[2]"));
            WebElement tableCellLastName = participantTable.findEle-
ment(By.xpath("//*[@id='participantTable']/tbody/tr[" + i + "]/td[3]"));
            WebElement tableCellBirthday = participantTable.findEle-
ment(By.xpath("//*[@id='participantTable']/tbody/tr[" + i + "]/td[4]"));
            if (participant(firstName).firstName.equals(tableCellFirstName.getText()) &&
participant(firstName).lastName.equals(tableCellLastName.getText()) && participant(first-
Name).birthday.equals(tableCellBirthday.getText())) {
                return true;
            }
        }
        return false;
    }

    @Then("the participants should be found in the overview")
    public void the_participants_should_be_found_in_the_overview() {
        for (Participant participant : participants()) {
            participantOverviewContainsParticipant(participant.firstName);
        }
    }

    @And("Patricia is on the participants overview page")
    public void patriciaIsOnThePatricipantsOverviewPage() {
        webDriver().navigate().to("http://localhost:8098/#/participant");
    }


    @When("Patricia opens {string}'s detail page")
    public void patriciaOpensDetailPage(String firstName) {
        webDriver().findElement(By.xpath("//*[@id=\"partici-
pantTable\"]/tbody/tr")).click();
    }
}
```

Appendix XI: Glue Code – Participant Registration StepDefs

```
package oq_glue_code;

import io.cucumber.java.en.And;
import io.cucumber.java.en.When;
import org.openqa.selenium.By;

import static oq_glue_code.TestContext.*;


public class ParticipantRegistrationStepDefs {

    @And("Patricia wants to register {string}")
    public static void  patriciaWantsToRegister(String firstName) {
        webDriver().findElement(By.linkText("Participant Registration")).click();
        addParticipant(firstName);
    }

    @When("Patricia enters {string}'s data")
    public static void patriciaEntersData(String firstName) {
        webDriver().findElement(By.id("firstName")).sendKeys(participant(first-
Name).firstName);
        webDriver().findElement(By.id("lastName")).sendKeys(participant(firstName).last-
Name);
        webDriver().findElement(By.id("birthday")).sendKeys(participant(firstName).birth-
day);
        webDriver().findElement(By.id("gender")).sendKeys(participant(firstName).gender);
    }

    @And("registers {string}'s data")
    public static void registersThem(String firstName) {
        webDriver().findElement(By.id("registerParticipant")).click();
        participant(firstName).id = Long.parseLong(webDriver().findElement(By.id("partic-
ipantId")).getText());
    }

    @When("Patricia registers {string}")
    public void patriciaRegisters(String firstName){
        patriciaWantsToRegister(firstName);
        patriciaEntersData(firstName);
        registersThem(firstName);
    }
}
```

Appendix XII: Glue Code – Participant Detail StepDefs

```
package oq_glue_code;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;

import static oq_glue_code.TestContext.participant;
import static oq_glue_code.TestContext.webDriver;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.CoreMatchers.not;
import static org.hamcrest.MatcherAssert.assertThat;


public class ParticipantDetailsStepDefs {

    WeightEntry participantBaselineWeightMeasurement = new WeightEntry();

    @Then("{string}'s details should be displayed")
    public void detailsShouldBeDisplayed(String firstName) {
        assertThat(webDriver().findElement(By.id("participantFirstName")).getText(),
is(participant(firstName).firstName));
        assertThat(webDriver().findElement(By.id("participantLastName")).getText(),
is(participant(firstName).lastName));
        assertThat(webDriver().findElement(By.id("participantBirthday")).getText(),
is(participant(firstName).birthday));
        assertThat(webDriver().findElement(By.id("participantGender")).getText(), is(par-
ticipant(firstName).gender));
    }

    @And("{string} has no baseline weight measurement entry yet")
    public void hasNoWeightEntryYet(String firstName) {
        webDriver().navigate().to("http://localhost:8098/#/participant/" + partici-
pant(firstName).id);
        assertThat(webDriver().findElement(By.id("weight")).getText(), is(""));
    }

    @And("Patricia wants to set {string}'s baseline weight measurement")
    public void patriciaWantsToRegisterBaselineWeightMeasurement(String firstName) {
        webDriver().navigate().to("http://localhost:8098/#/participant/" + partici-
pant(firstName).id);
    }

    @When("Patricia enters {double} kg in the weight field, {string} in the time field
and {string} in the comment field")
    public void patriciaEntersKgInTheWeightFieldInTheTimeFieldAndInTheCommentField(Double
weight, String dateTime, String comment) {
        participantBaselineWeightMeasurement.weight = weight;
        participantBaselineWeightMeasurement.dateTime = dateTime;
        participantBaselineWeightMeasurement.comment = comment;
        webDriver().findElement(By.id("weight")).sendKeys(weight.toString());
        webDriver().findElement(By.id("dateTime")).sendKeys(dateTime);
        webDriver().findElement(By.id("comment")).sendKeys(comment);
    }

    @And("she saves these entries")
    public void sheSavesTheseEntries() {
        webDriver().findElement(By.id("setBaselineWeight")).click();
    }
```

```
    @Then("{string}'s baseline weight entry should be found in the system")
    public void baselineWeightEntryShouldBeFoundInTheSystem(String firstName) {
        long id = Long.parseLong(webDriver().findElement(By.id("baselineId")).getText());
        assertThat(id, not(-1));
        //assertThat(webDriver().findElement(By.id("weight")).getText(), is(participant-
BaselineWeightMeasurement.weight));
    }

    @When("Patricia enters {double} kg and any valid date time")
    public void patriciaEntersKg(Double weight) {
        webDriver().findElement(By.id("weight")).sendKeys(weight.toString());
        webDriver().findElement(By.id("dateTime")).sendKeys("06-Jun-2020, 4:15pm");
    }

    @Then("she can set the baseline weight measurement")
    public void sheCanSetTheBaselineWeightMeasurement() {
        assertThat(webDriver().findElement(By.id("setBaselineWeight")).isEnabled(),
is(true));
        webDriver().findElement(By.id("setBaselineWeight")).click();
    }

    @Then("she cannot set the baseline weight measurement")
    public void sheCannotSetTheBaselineWeightMeasurement() {
        assertThat(webDriver().findElement(By.id("setBaselineWeight")).isEnabled(),
is(false));
    }

    @And("{string}'s baseline weight measurement is set")
    public void baselineWeightMeasurementIsSet(String firstName) {
        webDriver().navigate().to("http://localhost:8098/#/participant/" + partici-
pant(firstName).id);
        patriciaEntersKgInTheWeightFieldInTheTimeFieldAndInTheCommentField(68.5,
"15.5.20, 8:15am", "any comment");
        sheSavesTheseEntries();
        webDriver().navigate().refresh();
    }

    @Then("{string}'s baseline weight entry should be displayed on that page")
    public void baselineWeightEntryShouldBeDisplayedOnThatPage(String firstName) {
        assertThat(webDriver().findElement(By.id("weight")).getAttribute("value"),
is(participantBaselineWeightMeasurement.weight.toString()));
    }

    @Given("{string} did not give her consent so far")
    public void did_not_give_her_consent_so_far(String firstName) {
        webDriver().findElement(By.xpath("//*[@id=\"partici-
pantTable\"]/tbody/tr")).click();
        assertThat(webDriver().findElement(By.id("participantConsent")).isSelected(),
is(false));
    }

    @When("Patricia registers that {string} gave her consent")
    public void patricia_registers_that_gave_her_consent(String string) {
        webDriver().findElement(By.xpath("//*[@for=\"participantConsent\"]")).click();
        webDriver().findElement(By.id("updateConsent")).click();
    }
    @When("she displays {string}'s details")
    public void she_displays_s_details(String string) {
        webDriver().navigate().refresh();
    }
    @Then("Patricia should see on the participant detail page that the consent was
given.")
    public void patricia_should_see_on_the_participant_detail_page_that_the_con-
sent_was_given() {
```

```
        assertThat(webDriver().findElement(By.id("participantConsent")).isSelected(),
is(true));
    }
}
```

Appendix XIII: Glue Code – Helper Classes

Participant

```
package oq_glue_code;

public class Participant {
    public Long id;
    public String lastName;
    public String firstName;
    public String birthday;
    public String gender;
}
```

Weight Entry

```
package oq_glue_code;

public class WeightEntry {
    public Long id;
    public Double weight;
    public String dateTime;
    public String comment;
}
```

Appendix XIV: Audit Report

Betreff Audit Report BDD OQ Prototyp
Thesis Sabrina Leuenberger

Contents

wega Informatik AG | Aeschengraben 20 | CH-4051 Basel | Tel. +41 61 270 87 87 | info@wega-it.com | www.wega-it.com

1 Summary

Audit date: 22-JUN-2020 (10h-12h, 13h30-16h)
Attendees: Sabrina Leuenberger (auditee), Mathias Fuchs (partially)
Performed by: Evelyne Daniel (auditor)

1.1 Audit purpose

This audit is performed in the context of the thesis "BDD - A Practicable Approach for Computerised System Validation" from Sabrina Leuenberger, student in the University of Applied Sciences Northwestern Switzerland.
The goal of the thesis is the evaluation of test automation for OQs within a BDD (Behavior Driven Development) framework and in accordance to GAMP5 for a category 5 software. In the context of this evaluation, a prototype based on automation tools has been developed.
The purpose of the audit is to assess if the prototype incl. related documentation meets quality standards for GMPs quality systems (e.g. GAMP5).

1.2 Audit scope

- Thesis "BDD - A Practicable Approach for Computerised System Validation"
- OQ testing process using automation tools
- OQ execution for the JBA application v.1.1.0.0 (business application, test object for the prototype) using automation tools
- OQ test evidences, traceability and documentation

Out of scope of the Thesis and therefore of the audit:

- Document management (versioning, electronic signatures, template optimization, template handling, Approvals of executed test documents)
- Validation of the Testing Tools (Selenium, Scenarioo, Cucumber/Gherkin, Cucumber Selenium plug in, OQ Test App)
- SOP Software development with BDD (assumption: SOP covers change management, Set up OQ test app, Glue code versioning in sync with App version, code review of glue code, tester training).
- Test plan describing the entire verification process, related required roles, IQ process, explorative testing.
  Note: At the time of the audit, an "OQ rational" document was handed out to the auditor explaining several aspects relevant for the development of the test strategy especially regarding the risk based approach. As outcome of the audit it has been agreed to include these explanations into the thesis work.
- User requirements

1.3 Audit Process

The audit was performed as follow:

- Step by step explanation of the OQ Process considering BDD, set up and use of test automation tools by the auditee
- Review of the OQ documentation templates to be used for test execution & test review
- Review of the OQ documentation generated for the OQ of the JBA application v.1.1.0.0
- Additionally, in order to better assess the suitability of the process,

wega Informatik AG | Aeschengraben 20 | CH-4051 Basel | Tel. +41 61 270 87 87 | info@wega-it.com | www.wega-it.com

- Execution of the OQ test by the auditee using the "OQ Test app" on the JBA application v.1.1.0.0
- Review of the Test results in Scenarioo by the auditor using the Test Report form

1.4 Conclusion

Repeatable processes have been demonstrated for the OQ testing of GMP computer systems. Processes related to the definition, development, and validation, release, maintenance and retirement were not in scope of the audit. However especially aspects around definition (specification), release and maintenance were also considered where required for the OQ testing process.
Some minor deficiencies and recommendations have been identified, those are mainly needed to improve the robustness of the OQ process. These findings should not be regarded as being a comprehensive inventory of all existing GMP deficiencies; they represent only those deficiencies identified by the auditor However, the recommendation made have no major impact on system or process quality.
Overall the audited area meets GxP quality standards and demonstrates compliance to GxPs quality systems.

2 Classification of findings

- **Critical**
  A deficiency, which has produced, or leads to a significant risk of operating a system which is evaluated to present a major breakdown of a quality system element, including any finding that involves fraud, misrepresentation or falsification of products or data. Critical findings will compromise the success of inspections by health authorities / customers or could possibly lead to a recall of product.
- **Major**
  A non-critical deficiency which has produced or may produce a system / process that does not comply with GMPs and / or internal quality systems or which indicates a failure to carry out satisfactory procedures for the operation of a system. Major findings may potentially compromise the success of inspections by health authorities / customers.
- **Minor**
  A deficiency which cannot be classified as either critical or major, but which indicates a departure from current Good Manufacturing Practice, or applicable software procedures or industry standards
- **Recommendation**
  A recommendation is not considered a deviation from current Good Manufacturing Practice but represents a departure from best practice; represents an opportunity for process / quality improvement.

wega Informatik AG | Aeschengraben 20 | CH-4051 Basel | Tel. +41 61 270 87 87 | info@wega-it.com | www.wega-it.com

3 Audit checklist: Testing

Question	Answer
Do planned testing activities include unit, integration, system, and release testing? Please shortly explain all types of testing performed before release of software products.	Yes, focus here is on OQ, well explained in the "OQ rational" [Ref-1] (content to be included in the thesis, see note in 1.2 out of scope)
Is there written policy for testing process? If yes, please provide a reference (ID).	Yes, test specification v.1.1.0.0 [Ref-2] Note: Test specification is referring to test plan and SOP (which are out of scope of the thesis)
Does the testing procedures describe: • How test documents are developed and managed (reference) • What types and levels of testing are required? (reference) • How test outcomes are analysed for acceptability? (reference)	Yes, focus here is on OQ, see test specification v.1.1.0.0 [Ref-2] Note: Test specification is referring to test plan and SOP (which are out of scope of the thesis)
Do the written procedures describe which organizational groups are responsible for testing, reviewing, release, track of errors etc.? If yes, please provide a reference.	Written procedures in the test specification v.1.1.0.0 [Ref-2]
Does all testing documentation contain the information like Document Title and Version, or Accountability Signatures?	Principally Yes, test result v.1.1.0.0 [Ref-3], test report v.1.1.0.0 [Ref-4], feature files. Review approval feature files according to defined process: file consent_management feature is signed from SME and PO. See observation #6 Note: document management is out of scope of the thesis, see 1.2
Does test design documentation contain • References for traceability to the component's design and requirements specification? • Data and test equipment specification? • Criteria for acceptance/release of test component?	Yes, see feature file and test specification v.1.1.0.0 [Ref-2] and referring to test plan and SOP (which are both out of scope of the thesis, see 1.2)
Do completed test cases contain the following information: Documented test cases with defined inputs, expected outputs, and actual outputs, tester and eye witness if necessary.	Yes, see test specification v.1.1.0.0 [Ref-2] test result v.1.1.0.0 [Ref-3], test report v.1.1.0.0 [Ref-4], referring to test plan and SOP (which are both out of scope of the thesis, see 1.2) See observation #4

wega Informatik AG | Aeschengraben 20 | CH-4051 Basel | Tel. +41 61 270 87 87 | info@wega-it.com | www.wega-it.com

Question	Answer
Are automated testing tools in use? If yes, which tool and is that procedure validated? If so: for what tests and is it verified?	Yes, tools are used and validated. Validation of those itself is out of scope of the thesis. Only the use of it in the OQ context is part of the thesis and the verification process is well defined especially using the test results form to document the test set up and test execution. Automated test scripts should be controlled in accordance with a documented procedure. Feature files & glue code are described in the test specification v.1.1.0.0 [Ref-2]. Could be integrated in the "Software Coding Guideline SOP" in future. See observations #1, #2, #3, #5, #7

4 Observations and recommendations

#	Checkpoint	Observations	Classification
1	Specification document for JBA v1.1.0.0	The boundaries between "OQ test app" as part of the testing environment Tools and it's "configuration" in order to perform the OQ of the JBA application could be better described. It could also be described how the "OQ test app" is managed (change of glue code files). Validation of the Tools, see "assumptions and out of scope".	Minor
2	Specification document for JBA v1.1.0.0	Changes between JBA v1.0.0.0 and JBA v1.1.0.0 are not clearly documented. Recommendation: refer to a change document or describe the changes. Document why partial or why full review is needed based on the changes.	Minor
3	Result document for JBA v1.1.0.0	Documentation of integration of glue code in the "OQ Test app" not clearly stated (see observation #1). Segregation IQ app and set up OQ app to be better addressed (see observation #1).	Recommendation
4	Result document for JBA v1.1.0.0	Address clearly which steps are verification steps and which steps are execution steps e.g. set up of the "OQ Test app".	Recommendation

wega Informatik AG | Aeschengraben 20 | CH-4051 Basel | Tel. +41 61 270 87 87 | info@wega-it.com | www.wega-it.com

#	Checkpoint	Observations	Classification
5	The computer generated logs resulting from the execution of the automated test scripts are normally automatically from the execution of the scripts.	Yes, however the logs are not kept, only the files being then viewed in Scenarioo. Logs could be additionally kept or it should be documented why there is no added value in keeping those.	Minor
6	Review process	Location of files: test reviewer must perform some "setup" steps (unpack zip and save folder at right place) in order to be able to review test results in scenario. This is a Workaround that could be automated in future (OQ test app to create folder at right place).	Minor
7	Review process	Process is well defined, some improvements of the Test Report form can be done. Furthermore the use of the compare function in Scenarioo could be implemented in future.	Recommendation

5 References

#	Document
Ref-1	OQ Rational
Ref-2	Test Specification Version 1.0 – ID: Doc_JBA_v.1.1.0_TS_OQ
Ref-3	Test Results Version 2.0 – ID: Doc_JBA_v.1.1.0_TER_OQ
Ref-4	Test Report Version 2.0 – ID: Doc_JBA_v.1.1.0_TRep_OQ

wega Informatik AG | Aeschengraben 20 | CH-4051 Basel | Tel. +41 61 270 87 87 | info@wega-it.com | www.wega-it.com

Appendix XV: Test Specification

JBA Test Specification

Document ID: Doc_JBA_v.1.1.0_TS_OQ

According to GAMP5 Appendix D5 (ISPE, pp. 198-199)

Document History

Version	What	Who	When	Comment/Signature
1.0	New, issued for JBA v.1.1.0.0.	T'Challa, Test Analyst	15-Jun-2020	Digitally signed by Test Analyst (Author), SME (Reviewer), Process Owner (Approver)

Introduction

This document is based on the approved Test Plan. It is specifically dedicated to the OQ for JBA_v.1.1.0.0. As defined in the Test Plan, this document shall be written by the Test Analyst, reviewed by the SME and approved by the Process Owner. It describes the OQ part of the JBA v.1.1.0.0 verification process.

Purpose

This document defines the set-up and the rules to be followed for a valid OQ execution. It guarantees a consistent embedding of the OQ in the whole verification process. The goal of the OQ is to get the formal evidence, that the pre-defined functional specifications are met.

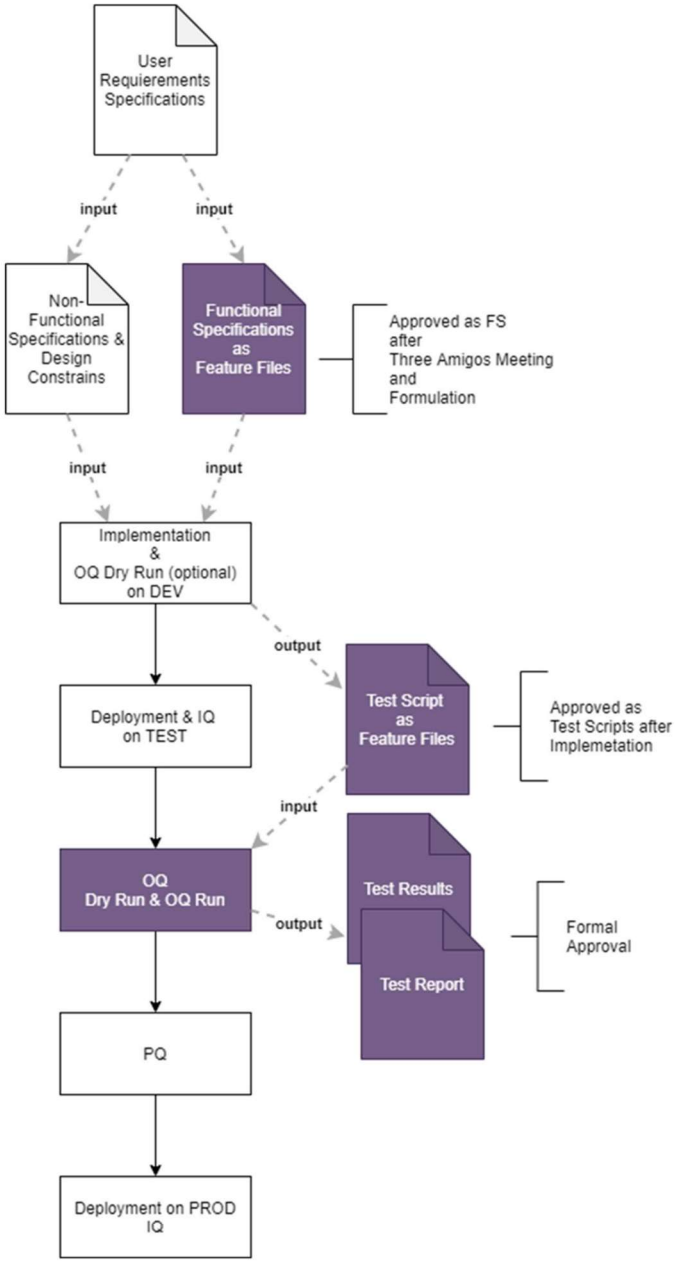


Figure 1: Overview of the whole verification process. The OQ part is highlighted in lilac

OQ Scope

Version of Software to be verified in this OQ:

- JBA v.1.1.0.0: It additionally includes the implementation of the feature file 'consent management'.

OQ tests to be performed, required test review and reason:
OQ tests with a full test review to be performed on following feature files:

- Consent management, signature Version 1.0.0.0 approved by the Process Owner the 15th of June 2020 (new feature)
- Setting the baseline weight measurement, signature Version 1.0.0.0 approved by the Process Owner the 4th of June 2020 (to verify correct integration, business critical)

OQ tests with a partial test review to be performed on following feature files (regression only):

- Registration of a participant, signature Version 1.0.0.0 approved by the Process Owner the 4th of June 2020
- Participant's overview, signature Version 1.0.0.0 approved by the Process Owner the 4th of June 2020

Personnel/Resources

Following roles and persons have an active role for the execution of OQs:

- Test Analyst: T'Challa, ct
- Process owner: Hank McKoy, mh
- SME: Patricia Walker, wp
- Tester: Scott Lang, ls
- Test Reviewer: Patricia Walker, wp
- Quality Unit: Hope Pym, ph

Methods

The OQ of the JBA application will be done by the test automation software 'OQ Test App'. The OQ is done in an automated way based on the feature file (test scripts) and the glue code (automated tester).

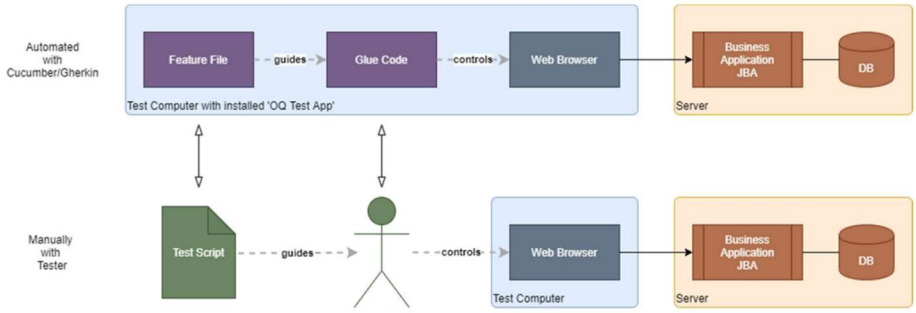


Fig. 2: Analogy between manual and automated testing.

In order to assure and confirm the correctness of the tests, a test review will be performed in a similar way as it would have been done for testing performed by a human tester.

The review is then based on the Test Results displayed in Scenarioo and documented on a report. The process shown in the Annex will be followed.

Prerequisites
Before the OQ shall be performed following prerequisites needs to be fulfilled:

- IQs on JBA v.1.1.0.0 successfully performed
- OQ Test App validated
- Scenarioo validated

Environment
JBA Test Environment Platform.

Tools (including automated test tools)
Following tools have to be used:

- OQ Test App: To perform the OQ testing in an automated way
- Scenarioo: To display the test results for review and approval

Reference to Specifications
As the feature file is a combined functional specification and test script document, only the references to the user requirements are needed. The user requirements are identified by their unique ID, which is noted in the "Specification brief" section of the feature file.

Required Documentation
Following documentation are required for OQ:

- Feature Files: Written by the Three Amigos or a delegate, reviewed by the SME and approved by the Process Owner.
- Glue Code: Written by the supplier (incl. a code review), 2nd code review by the Tester on the changed parts of the Glue Code.
- Test Result: consists of the autogenerated test results by the OQ Test App and the filled in document Doc_JBA_v.1.1.0_TER_OQ version 1.0: reviewed by the Test Reviewer, approved by the QA.
- Test Report based on Doc_JBA_v.1.1.0_TRep_OQ version 1.0: filled in by the Test Reviewer, reviewed and approved by the QA

Test Execution
The execution of the tests includes setting up the OQ Test App, running the automated runs (first as a dry run, then as a validation run) and filling out the form Doc_JBA_v.1.1.0_TER_OQ version 1.0. A zip file of the test results is placed in the folder \\dia\InterneProjekte\BDD4OQ\JBA\v.1.1.0.0\OQ\TestResults.

Test Review
The test results will be reviewed and based on that review a report will be written.
To guide that process the document „Test Report form" Doc_JBA_v.1.0.0_TRep_OQ version 1.0 has to be used, filled in and signed by the Test Reviewer.
In order to view the test results in Scenarioo, the zip file found in:
\\dia\InterneProjekte\BDD4OQ\JBA\v.1.1.0.0\OQ\TestResults is taken and checked if the signature of the tester is valid. The zip-file is unzipped and the folder with the test results is put into the location:
\\dia\InterneProjekte\BDD4OQ\Scenarioo\Reports\JBA Version 1.1.0.0
Scenarioo can then be started and the results viewed in the Chrome browser with following url: http://localhost:8080/scenarioo/
The filled-in and signed form Doc_JBA_v.1.0.0_TRep_OQ should be placed in following folder for approval:
\\dia\InterneProjekte\BDD4OQ\JBA\v.1.1.0.0\OQ\TestReview

Two types of test reviews are foreseen:

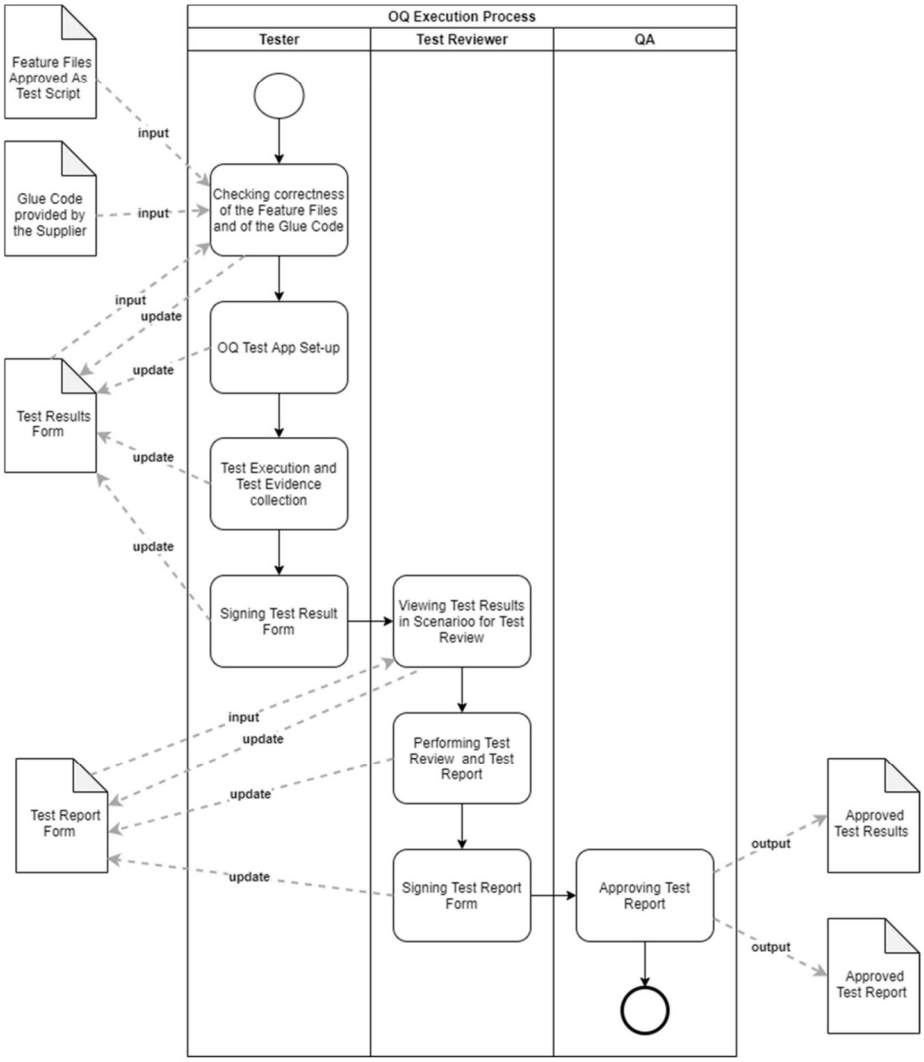
- *Full Test Review*, for which each step with screenshot and the other available data needs to be verified: The goal is to fully verify the new functionalities and their integration. Therefore, the full test review is foreseen for all tests defined in new feature files, feature files that where subjected to changes since last OQ or for feature files that prove the correct integration of the new functionalities into the already existing JBA v.1.0.0.0. If whished by the SME or the Process Owner some additional feature files could also be subjected to the full test review (e.g. they might be of special business criticality).

- *Partial Test Review*, for which the step by step verification can be omitted, as it concerns functionalities that did not change (regression tests) and for which it has been proven in a former OQ process, that the OQ Test App is working correctly and as expected[1].

Test Approval
The Doc_JBA_v.1.1.0_TRep_OQ that was filled and signed by the Test Reviewer is reviewed and approved by the QA.

[1] This allows that the complete specification suite is tested in the OQ of each new version, which would be very time consuming if performed manually.

Annexe: OQ Execution Process



Appendix XVI: Form - Test Results

JBA Test Execution: Results

Document ID: Doc_JBA_v.1.1.0_TER_OQ

Based on GAMP5 Appendix D5 (ISPE, pp. 204-205)

Document History

Version	What	Who	When	Comment/Signature
1.0	New template for OQ of JBA v1.1.0.0	T'Challa, Test Analyst	15-Jun-2020	digitally signed by Test Analyst (Author), Test Reviewer (Reviewer), QA (Approver)

Purpose

This document should allow the tester to document his activities and the reviewer to review them. It is conceived as check list in order to guide and support the tester while preparing and performing the automated test.

Tested Version

JBA_v.1.1.0.0.

Feature Files

Verify the following:

- ☐ The last digital signature for approval of each feature file is valid (last line of the feature file history).[1]
- ☐ Correct 'Sig. Version' of each feature file is available (compared to test specifications: Doc_JBA_v.1.1.0_TS_OQ)

Glue Code[2]

Verify the following by performing a code review of the glue code covering a complete check of all the changes in the glue code compared to version 1.0.0.0:

- ☐ The glue code with version 1.1.0.0 is available
- ☐ The changes in the glue code compared to version 1.0.0.0 only uses Selenium methods to control the test automation (directly or indirectly by calling another StepDef method)
- ☐ Selenium actions which a human tester would not perform in the same way, were not found in the glue code. For example the function *webDriver().navigate().refresh();* should only be found for steps to control or establish the prerequisites or when a tester would also be asked to refresh a page, e.g. after saving.

[1] From the moment on that the feature files are approved as FS initial version, they will be developed and maintained in parallel with the JBA code. In order to go live: the approved TS that is ready for OQ and digitally signed will be committed and pushed to the git repository in the same way as for code. It will be released together with the glue code and the JBA code with the same release version number. In order to assure, that it has not been changed during that process the tester needs to verify the digital signature.

[2] The glue code and the JBA code are developed and maintained in parallel, meaning they have the same release version and are submitted to the same code reviews process. The code review performed herewith by the tester is therefore the second review.

- ☐ The changes in the glue code compared to version 1.0.0.0 only uses hamcreast matcherassert methods in order to compare the result with the expected results to make a test step passed or failed
- ☐ The changes in the glue code compared to version 1.0.0.0 do not use the rest api to the backend to perform the tests. The rest api might be used in the Before - or After Step definitions in order to perform pre or post condition test steps.
- ☐ No StepDef method is empty (only to be checked for the changed parts in the glue code).
- ☐ If given:
  - ☐ Interfaces to peripheral systems are only used for assertions. The JBA function under test must be triggered using the UI by Selenium

OQ Test App Set-up

Do the following:

- ☐ Integrate Feature Files in the OQ Test App
- ☐ Integrate Glue Code in the OQ Test App
- ☐ Perform dry run:
  Test ID: _____
  Verify the following:
  - ☐ All feature files were run
  - ☐ All scenarios were performed
  - ☐ All steps gave a result (either passed or failed)

Test Execution

Test ID (*Example Build-2020-04-17-03-00-00*): _____
Test Date: _____
Starting Time: _____
Observations: _____

Test Evidence

Test Result folder as zip file into this section:

Test Summary

By signing this document, I confirm the careful execution of the above-mentioned activities. The automated tests have been carried out correctly and diligently to the best of my knowledge and conscience.

Appendix XVII: Form - Test Report

JBA Test Report

Document ID: Doc_JBA_v.1.1.0_TRep_OQ

Based on GAMP5 Appendix D5 (ISPE, pp. 205-206)

Document History

Version	What	Who	When	Comment/Signature
1.0	New template for OQ of JBA v1.1.0.0	T'Challa, Test Analyst	15-Jun-2020	digitally signed by Test Analyst (Author), Test Reviewer (Reviewer), QA (Approver)

Purpose

The purpose of this report is to document the review of the test results and to assess the findings, if any. The outcome of this process is a decision on the success or failure of the OQ and the measures to be taken, if needed.

Review Initialisation

Following items have to be checked:

- ☐ The test results and the accompanying document Doc_JBA_v.1.1.0_TER_OQ were performed, filled in and signed by the Tester
  Test ID: _____
- ☐ The test ID, as noted in Doc_JBA_v.1.1.0_TER_OQ corresponds to the test ID of the test found in \\dia\InterneProjekte\BDD4OQ\JBA\v.1.1.0.0\OQ\TestResults embedded in the Test Result file
- ☐ The Doc_JBA_v.1.1.0_TER_OQ was filled in and signed by the tester and the document is complete
- ☐ Review of the test results in Scenarioo is possible: Scenarioo can be started and the branch JBA Version 1.1.0.0 is visible
  Displayed Build (*Example Build-2020-04-17-03-00-00*): _____
  Description if review is not possible: _____

Review Test Results

Following points have to be checked in Scenarioo on the table of the 'Use Cases'-Tab:

- ☐ All feature files as described in the test specification (Doc_JBA_v.1.1.0_TS_OQ) show up in the 'Name'-column
- ☐ All scenarios were performed (to verify, click on every row of the table and check for each, that a table is displayed with a list of scenarios. The number of the listed scenarios should correspond to the 'active scenarios'-number as found on the last line of the 'Long Description' section on the right).
- ☐ Test results: 4 success, 0 failed

1) Full Test Review

This test review checks the test coverage and the test results based on the feature files as defined in the test specification (Doc_JBA_v.1.1.0_TS_OQ).
Features with full test review (list the features, as displayed in the 'Name'-column' of the table found in the Scenarioo 'Use Cases'-Tab, for which you performed the full test review):

- _____
- _____

Review Test Results:

Following points have to be checked:

- ☐ All steps were performed in the tests (to verify, click on every row of the table found in the Scenarioo 'Use Cases'-Tab and check for each, that the sum of the numbers in the '# Steps' column of the displayed table corresponds to the 'active steps'-number as found on the last line of the 'Long Description' section on the right)
- ☐ Screenshots show accurate test results
  - ☐ Deviations are described in the table 'Test Deviation' underneath in the chapter 'Test Report'
- ☐ No test errors detected based on the results displayed in Scenarioo
  - ☐ No errors detected in the scenario/step descriptions
  - ☐ No errors detected in scenario/step execution
  - ☐ Errors are described in the table 'Test Deviation' underneath in the chapter 'Test Report'
- ☐ For all steps: step_duration > 0 s 0 ms
  - ☐ Deviations are described in the table 'Test Deviation' underneath in the chapter 'Test Report'
- ☐ Acceptance criteria (then-steps) are fulfilled
  - ☐ Deviations are described in the table 'Test Deviation' underneath in the chapter 'Test Report'
- ☐ Failed Tests:
  - ☐ All records to evaluate test failures are available
    - ☐ Deviations are described in the table 'Test Deviation' underneath in the chapter 'Test Report'
  - ☐ Failed Tests are described in the table 'Test Deviation' underneath in the chapter 'Test Report'

Additional comments (if any, else enter n/a): ____

2) Partial Test Review

This Test Review checks the test coverage and the test results based on the feature files as defined in the Test Specification (Doc_JBA_v.1.1.0_TS_OQ).
Features with partial test review (list the features, as displayed in the 'Name'-column' of the table found in the Scenarioo 'Use Cases'-Tab, for which you performed the full test review):

- _____
- _____

Review Test Results:

Following points have to be checked:

☐ For all use cases, results show success
☐ Spot check of test results do not show any errors or deviations
Additional comments (if any, else enter n/a): ____

Test Report

The test report section is based on the test results and the test review as performed above. In the test report section, an evaluation of the severity of the failed steps, if applicable, and an overall evaluation of the OQ success is performed.

Deviations:

Fill in following checks, and if given the table underneath:

☐ Not applicable
☐ Applicable, see following table:

ID	Feature Scenario	Step	Error / Deviation Type	Risk Assessment	Evaluation	Decision
01	*Feature and Scenario Name*	*Step of the scenario, e.g. 3rd step (And)*	Error in the Application	Data integrity cannot be guaranteed with probability = high	Risk is to high, that go-live would be possible.	Change request: bug bddoq-78

Conclusion(s):

____

Appendix

Test results on which this report is based are put as zip file into this section:

Summary

By signing this document, I confirm the careful execution of the above-mentioned activities. The test review has been carried out correctly and diligently and the test report has been written to the best of my knowledge and conscience.

Appendix XVIII: JBA User Stories

JBA User Stories

Setting Study Parameter (BDDOQ-11)

BDDOQ-20
In order to get a meaningful series of weight measurements
As the clinical trial project leader
I want to be able to set the weight measurement frequency before the start of the study.

Registration of Participants (BDDOQ-12)

BDDOQ-21
In order to assign the measurements to a participant
As a nurse,
I need to be able to register the new participant.

BDDOQ-51
In order to immediately set the baseline measurement
As a nurse,
I need to be redirected from the Participant's Registration page to the Participant's page.

Consent Management (BDDOQ-13)

BDDOQ-22
In order to be able, perform the baseline measurement and the next measurement in conformity with that participant's will
As a nurse,
I need to enter into the system, when the participant has given his consent.

BDDOQ-23
In order to respect the participant's will, not to continue any longer in the clinical trial
As a nurse,
I need to be able to register it, when a participant withdraws his consent.

BDDOQ-24
In order to be make sure to be compliant,
As the process owner,
I want the system to control if the participant's consent was given or was retrieved.

Setting Participant's Baseline Parameter 'Weight' (BDDOQ-14)

BDDOQ-26
In order to set the baseline of the measurement series of a participant
As a nurse,

I need to register the baseline weight measurement.

BDDOQ-50
In order to keep track for which participants the baseline weight measurement is missing,
As a nurse,
I need to have an overview of the registered participants and the information, if the baseline weight measurement was already set. From this overview, the details of each participant should be reached, e.g. to set the baseline weight measurement.
Acceptance criteria:

- An overview over all registered participants is available, where the first name, last name, birthday and gender can be found for each participant
- From this overview it is possible to reach each of the participant's page, where the baseline weight measurement can be set
- It is possible to determine in the overview, if for a participant the baseline weight measurement was already set or not
- It is possible to get a subset of the participants with the same gender or the same status in respect of the baseline weight measurement setting (baseline weight measurement is set: yes/no)
- The overview can be sorted according to the last name of the participants

Notification for next Weight Measurement (BDDOQ-15)

BDDOQ-27
In order to not forget to make a new measurement
As a nurse,
I want to be reminded, when a new measure has to be done on a participant.

Registration of next Measurement (BDDOQ-16)

BDDOQ-28
In order to register the next measurement done on a participant
As a nurse,
I need to be able to enter the result.

Reporting (BDDOQ-17)

BDDOQ-29
In order to be able to evaluate the results of the participants weight measurements
As the clinical trial project leader
I need to be able to get the participant's data.

BDDOQ-30
In order to see the effect of the potential new treatment on the participant's weight
As the clinical trial project leader
I want to get a result chart of the weight measurements for a participant.

User Management (BDDOQ-18)

BDDOQ-31
In order to manage the access of users to the system
As the system administrator
I need to manage access of people to the system and assign roles to users.

BDDOQ-32
In order to minimise the risk of unauthorised access to the system
As the system administrator
I want to be able to block the access of a user that was not active on the system for more than three months.

Data Integrity (BDDOQ-19)

BDDOQ-33
In order to be able to check data integrity issues
As the system administrator
I need to be able to see who changed which data when and in what way.

Reference to the Project Repository

The complete project documentation and the source code can be found in the GitHub Repository BDD4OQ under the tag 'Thesis BDD4OQ - JBA v.1.1.0'

https://github.com/sableu/BDD4OQ/tags