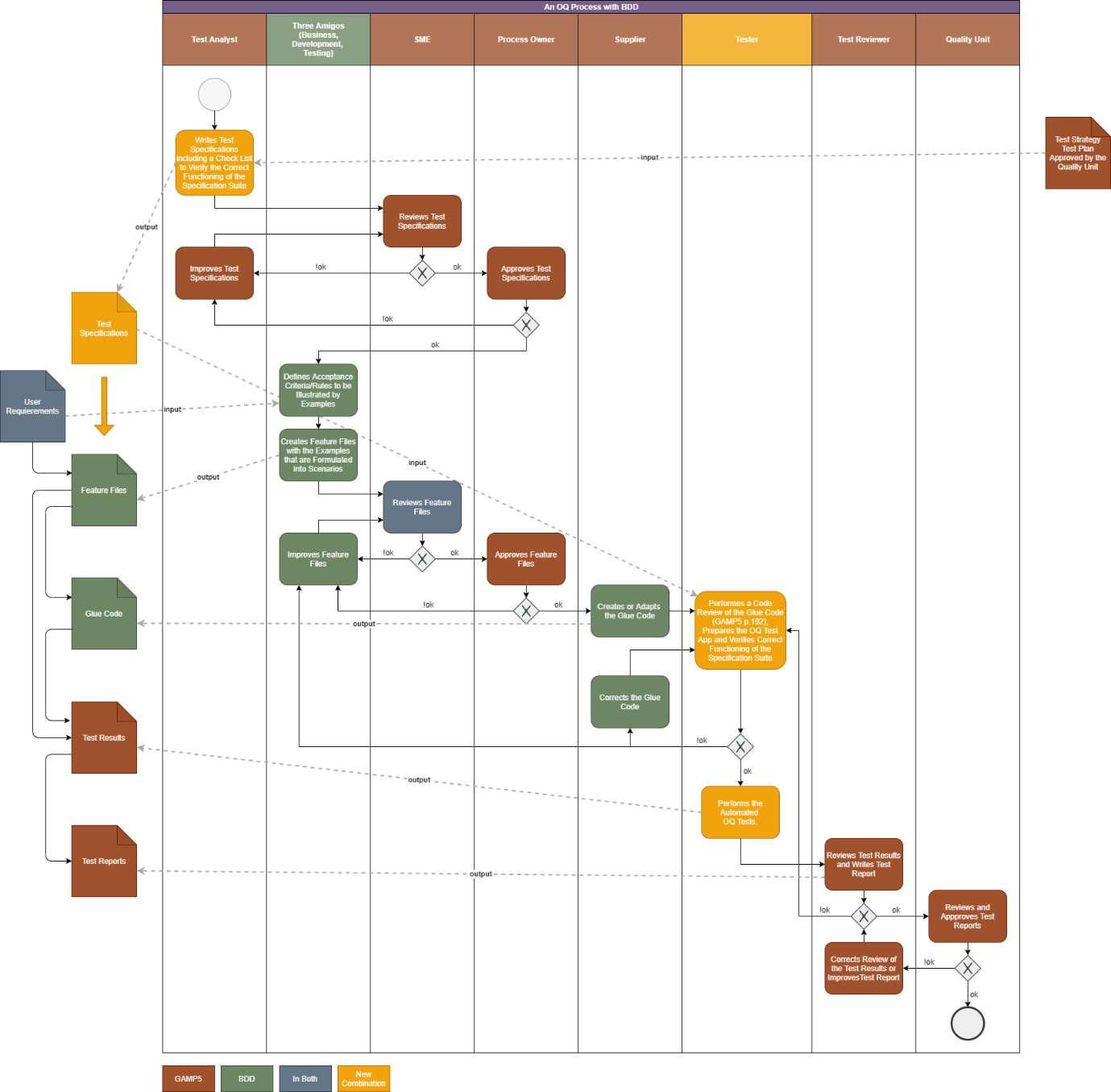
**OQ Rational**

The OQ and the associated documents are based on certain basic assumptions which will be explained in the following:

**Test Plan**

The test plan was not written, as it is out of scope in respect to this project. The test plan would define, among others, a high level OQ process to determine the outline of the procedure and the responsabilities. This is important, as it determines who has to write the test specification and which guidelines has to be followed.

Therefore, following process was used as a basis:



The OQ process as defined in a test specification has one difference: The recursion loops foreseen after the Tester gets the feature files and the glue code were left out for following reasons:

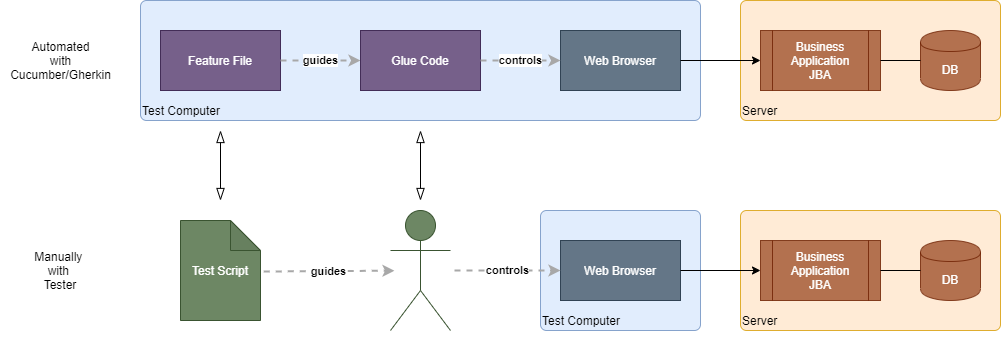
* The test is rapidly done. To stop it and clarify some points or make some decisions before continuing testing does not make sense in the set of an automated test.
* In the set-up as defined in the test specification, a good way to give feed-back and to improve the test would be to fill in the form as also foreseen for going through the complete process.
* As explained in the next chapter, the OQ process has the goal of performing the OQ on the JBA, but at the same time, to check the correct behaviour of the glue code and the feature files. Therefore, it also makes sense to go through the complete process without recursion.
* All findings will be documented and at the end of the process it will be decided what actions to take (happy path 🡪 no findings initiate PQ)

The other processes where introduced in the test specifications for sake of clarity. I suppose, that normally you would not add them in a test specification.

**Glue Code Validation and Feature File Check**

As for any code, it needs to be verified that the glue code does what it should do.

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



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

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

GAMP5 foresees that IQ, OQ and PQ have to be performed on applications in order to verify its fitness for intended use:

* IQ assures, that the right version is installed and used 🡪 version control of the glue code: For each version of a feature file a corresponding version of the glue code needs to be available/used. This could be done using Git, which one of its main functionalities is the version control of the code. This assures, that it can be traced unambiguously.
* OQ assures, that the functions are performed correctly and as intended.
* PQ is not applicable as there is no functionality of the glue code that addresses a human (no user acceptance testing, as there are no human users). The role of the human is to make sure, that the glue code makes it job, which brings us back to the OQ.

[Remark: During the audit it turned out, that this is confusing 🡪 glue-code is not an application on its own 🡪 no need for IQ, OQ and PQ]

GAMP5 further states, that the verification can be done by testing, but it would also be possible to use other methods (GAMP5, Table 4.1, p. 356).

To define an applicable method to verify the glue code, the analogy to the tester could be used again: The tester is also error prone (possibly even more than the glue code, even not fully tested). Therefore, and while performing the tests, the tester needs to take screenshots and write some descriptions as testing evidence so that during the subsequent test review, the performed tests can be verified. This same method could also be used to verify the correct functioning of the glue code. Therefore, Scenarioo is used to display a test description, some data for evidence on the test performance and screen shots. If one could fully rely on the glue code the standard built-in cucumber test report would be sufficient.

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

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

One additional check could be done, but this would be of less relevance as such practice would also be detected sooner or later on the screenshots and if such practices would be used by the supplier, it could be considered as fraud (not anymore as error): Additional but unnecessary selenium steps, such as the reload/refresh of a page or a not foreseen save action, could be introduced to voluntarily hide some specific behaviours of the app. To avoid this, it should be addressed in the contract with the supplier and it should also be checked by the tester by verifying that not necessary functions of selenium were introduced. But at latest in the PQ or during some additional explorative testing, if there are some hints, that the supplier acts fraudulently, such practice would be detected.

* Prerequisite Tester needs to know Selenium.