

Verify and validate a solution

Business Analysis

Analysis Skills : 3

Do not confuse verification and validation of a project. The first ensures that the design has been well conceived and carried out without error, while scrupulously following the specifications. The second ensures that the solution implemented meets the basic problems.

Published Wednesday July 21st 2021, 15:02
Modified Monday August 26th 2024, 10:04



By Olivier Paudex

Introduction

The execution of a project goes through various stages that range from planning to solution integration, through needs analysis and requirements management. But once the latter is delivered, you might think that the BA's work stops there. But this is not the case, the BA will actively participate in the verification and validation of the implemented solution with what has been defined in the requirements. Here's a little overview of the tasks.

The SE (Solution Evaluation)

The **SE** is the section about evaluating a solution, analyzing its performance, and making recommendations that can improve its added value as seen by the **BABOK**.

This publication is going to address testing throughout a project.

A **PDF** attached to this publication, in the form of a data stream, summarizes all the tasks, their descriptions, tools, stakeholders involved, and mnemonic techniques. It has been produced for educational purposes.

Knowing the difference between the two tasks

Verification is about ensuring that the design is well thought out and error-free. Validation, on the other hand, ensures that the implemented solution meets the basic need or problem.

The two tasks go hand in hand. The cycle often starts when the customer asks to solve a problem. The project team will then define and verify that the discovered requirements solve the problem. Questions to ask might be :

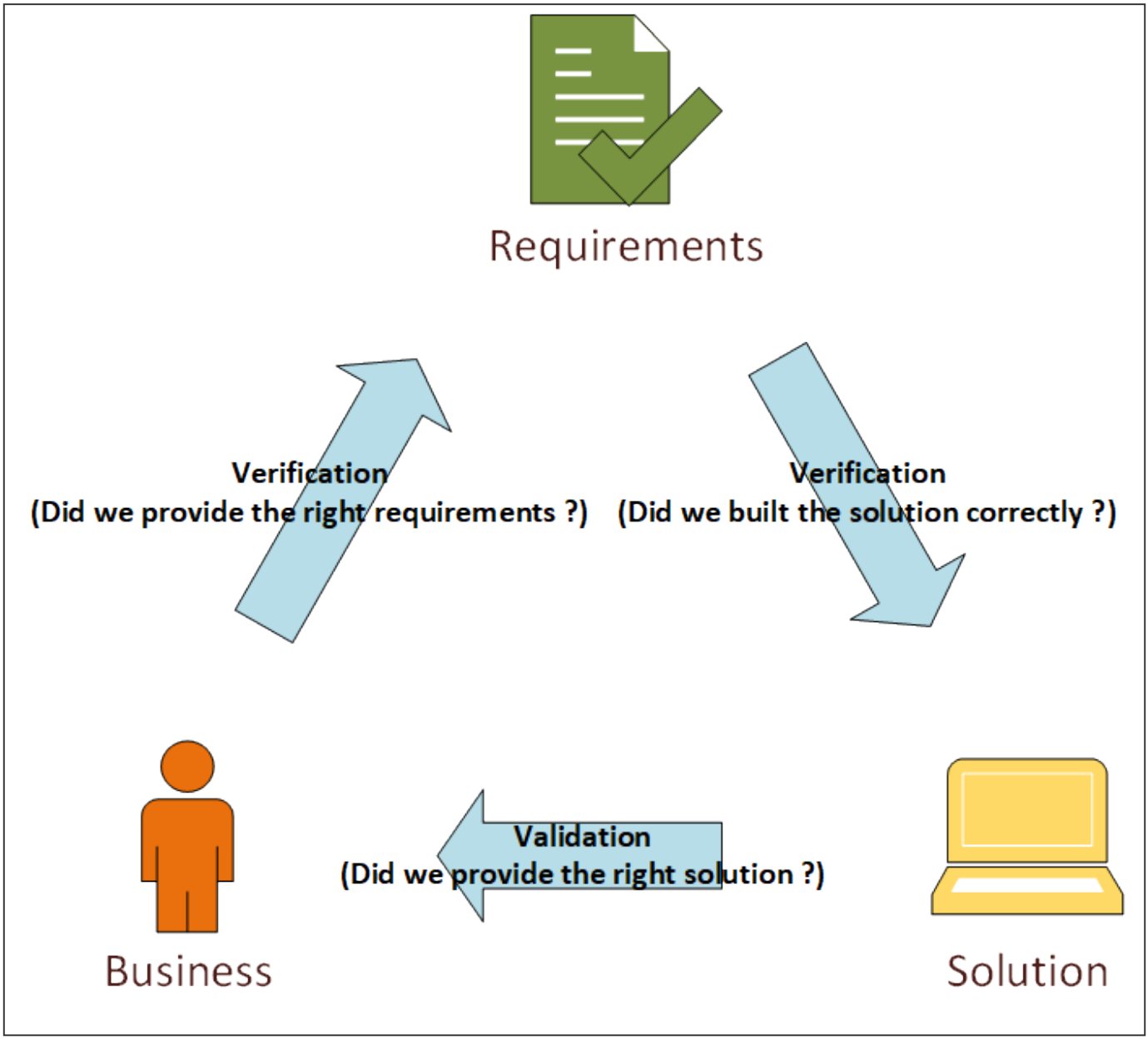
- **Did we provide the right requirements provided ?**
- Are the deliverables appropriate ?

Then we move on to developing a solution and when checking the solution, the question to ask might be :

- **Did we built the solution correctly ?**
- Did we solved the main problem (root cause) ?

Then comes a whole phase of testing, fixing, until the client **validates** the solution. The question would then be :

- **Did we provide the right solution ?**
- Does the solution provided work ?



The verification and validation cycle

When does the testing phase begin

Testing is done as early as the planning phase by adding an estimation of the time needed to perform it.

Then in the research phase, during stakeholders elicitation and writing the requirements in the **backlog**, a quality assurance team will review them and validate if they are testable. The BA is responsible for describing a whole arsenal of test cases to be performed. The testing ends with the final acceptance of

these tests by the customer. This is called, **a UAT (User acceptance test)**.

The four phases of testing

The smoke test

It verifies that the solution has no apparent major flaws. It reveals simple errors that would prevent the other three test phases from being executed.

This definition comes from the field of electronics. When a new machine was powered up for the first time, designers checked that it did not **"smoke"**, in the literal sense.

The unit test

This is the second phase of testing, after the relatively simple smoke test. It is a test performed on a module or a small part of the solution.

For example, software is broken down into functions that are tested one after another.

The advantages of subjecting modules to unit testing are as follows :

- Until a module passes the unit test, it will not be able to be submitted in the global integration test. For example, we'll test a car's engine unloaded, before mounting it on the chassis and checking that it can run.
- It is entirely possible to prioritize the testing phases. To use the car example, it is entirely possible to verify that certain engine components work together before building the entire engine and integrating it into the chassis.
- When a module is operated by an interface, it is recommended that it is verified by multiple people. Every human reacts and thinks differently. Doing so avoids errors that the designers would not have seen only because there was nothing to suggest that someone would operate the interface in a different way.

The global integration test

The third phase of testing consists of bringing together all the modules tested in the second phase and testing their operations together in a dedicated environment. This is called a **"Sandbox"**.

The test system

This is the fourth and final phase. It is also the last chance test left to the designers before the hand passes to the users for the final acceptance phase (**UAT**).

This test will confirm that the solution covers the original problem. It is categorized into several subcategories including :

The validation of requirements

This test verifies the logic of the solution by ensuring that the requirements analyzed throughout the project are met.

The regression test

Here, it is about being able to re-test all changes made from a certain point. This check ensures that a last minute change does not interfere with what was previously working.

The dynamic test

This test runs the solution must work under different circumstances. These include :

- **The performance test**. This demonstrates how fast the solution should respond. It should match the response time defined in the non-functional requirements.
- **The stress test**. This demonstrates the limitations of the solution. An engine is not going to respond the same way on a highway as on a mountain road.
- **The volume test**. This demonstrates how well the system works if it has to process a huge amount of data. It is done in anticipation of the future.

The security test

Here, it is nothing more or less than managing access rights in a software or functions of a solution that would have several levels of use.

The installation and configuration test

This test ensures that the software will install correctly on different platforms (version, minimum memory, etc...), or run on different browsers.

The usability test

This test asks for user feedback on how the application is used. We're looking for users' opinions on how easy the interface is to use.

Here are some tips when creating interfaces :

- **Always display the system status**: Integrate a progress bar into the interface.
- **Write in user-understandable vocabulary**: Make the vocabulary within reach of everyone.
- **Add options to cancel and repeat the last task performed** Make sure the user always has an option to cancel or repeat the last task performed.
- **Be consistent**: Always use the same words, colors, symbols.
- **Provide drop-down lists**: Add choice lists, rather than having the user type in information. This prevents typos.
- **Check input immediately**: Check, for example, that a phone number is all numbers, but no letters.
- **Provide time-saving shortcuts**: Think about experts or users who will have to enter the same information a lot of times.
- **Create an intuitive and simple interface**: Don't add unnecessary information.
- **Create information bubbles to guide the user through the process**: Think about adding help to guide the user through the process.

Defining test cases

Test cases are step-by-step instructions. To create a good test instance, you need, among other things :

- Describe precisely what the tester should perform.
- Define what the tester should enter as data and what it should have as a result.
- In what environment the test should be performed (OS, Version, etc...)
- Describe any special procedures or requiring proper handling.
- Document the priority of tests. Should one test be performed before another.
- Ask the tester to approve the result.

Defining a test plan

The plan is going to describe all the verification and validation steps. It's going to describe how a solution should be verified. Here are some tips :

- Explain the purpose of the project and what the solution is going to be subjected to.
- Define the environment in which the tests are to be performed.
- Define the cases that are going to be tested.
- Define the data needed for the test.

Case to test	Expected result
Purchase a taxed product	The tax must be added to the amount of the product at the time of purchase
Purchase a non-taxed product	The amount of the product is calculated without adding a tax
Buy several products, some taxed, some not	The tax must be added only to the amount of the taxed product

Description of the features to be tested

- Define the test cycle (time between each test) and the tools to be used.
- Define the pass/fail criteria.
- Define what to do if a test fails.
- Determine a table of testers and responsibilities.

Conclusion

The testing phase is critically important. It ensures that the implemented solution works properly according to the requirements provided in the backlog. The BA's task will be to define a test plan, in which all cases to be tested will be described. It is recommended to break the test plan into several parts and to perform unit checks before testing the whole solution. It is also recommended to create a test platform (**staging**) or a risk-free environment (**sandbox**). A quality assurance team will then execute the test cases. Finally, end users will need to validate the plan by signing it.