Test-planing

Pizza Dronz is a small Maven-based project, and the flexibility of Agile development makes it particularly well suited to this type of project. Agile development emphasises rapid response to changes in user requirements, which is especially important for small projects that need to iterate and make improvements in a short period. For example, the Pizza Dronz project needed to read orders from a REST service and use an A* algorithm to figure out the best delivery path, while also avoiding no-fly zones. These requirements may change over time, for example, new no-fly zones may appear, or the calculation of delivery paths may need to be optimised. Agile development can respond quickly to these changes in requirements.

Test-driven development is a part of agile development that emphasises building and executing tests before implementing code or system components. In addition, it improves the outcome of the system by ensuring that the system implementation meets its requirements.

As a result, we typically explore when and how to conduct tests to ensure that each functional and non-functional (e.g., correctness, performance, efficiency, robustness and safety) requirement is met. We will delve into the specific requirements and, in particular, what support facilities and tools need to be used. In addition, we will look at how all testing processes can be integrated into the agile development lifecycle.

Early test design is closely related to Agile Test-Driven Development. In Agile Test-Driven Development, we first write tests and then write code that passes those tests. This approach emphasises the importance of early test design and makes it a core part of the development process.

During the early testing phase, unit tests help us catch bugs ahead of time. By writing unit tests for each function or method, we can ensure that the code is correct from the very beginning of its implementation. In this way, any bugs or problems can be detected and fixed at an early stage of the development process, thus avoiding costly bug fixes at a later stage of development.

Therefore, early test design not only improves the quality of software, reduces development errors, and improves development efficiency and effectiveness, but it is also a key component of agile test-driven development.

Test Procedure:

Unit Testing: It is crucial to conduct unit tests during the development process. Each function or method should have corresponding unit tests to ensure that they are correct. These tests should be done while the code is being written so that problems can be found and fixed early. If the unit tests are valid, the main errors that still need to be found in the integration tests are interface errors.

Integration testing: Integration testing should be performed when a feature is complete. This means that after all the components

that make up the feature have passed the unit tests, we need to test how these components work together. The testing effort should focus on the interfaces between units rather than their internal details. For example, for the central area and no-fly zone requirement, we first need to read valid orders from the REST service within a specified number of days, and then we need to test how the drone can find the best path to the central area while avoiding the no-fly zone.

System testing: when the entire application is complete, system testing is performed. This involves testing the entire system for non-functional requirements such as performance, usability, security, and the completeness and correctness of all functional requirements.

If at any stage the application does not pass a certain level of testing, it should be corrected and the tests should be run again. This process should be repeated until the tests pass and thus the requirements are met. This is because we use test-driven design in the Agile lifecycle, which emphasises building and executing tests before implementing code or system components.

If a bug is discovered during the design phase or unit testing phase, the cost of fixing the bug is relatively small. However, if a bug is discovered during the integration test phase or system test phase, the cost of fixing the bug is significantly higher. At the unit level: a thorough test is being used to validate the following information:

- The order number is 8 hexadecimal string.
- A valid date and year of the order date.
- The card number must be 16 numbers.
- The expiry date of the card should be in the format: MM/YY.
- Users can place an order on the last day of the expiry date, but cannot place an order if the expiry date is exceeded。
- The CVV must be 3 digits.
- The users are only allowed to place 1 to 4 pizzas in the same restaurant.
- The pizza names must match the menu of the restaurant.
- The total price is equal to the total price of pizzas and 1 pound delivery fee.
- Every restaurant has different open and closed dates and the user can only place the order when the restaurant is opening.