System Analysis & Design

Planning and Implementing an Object-Oriented Software System

Assignment Compiler: Michael Weiss

Aline Silva Rabelo
2019438

Francisco Leite
2019300

Summary

Introduction

The objective of this project was to implement a new film rental kiosk system for the company Xtra-Vision. All the project was produced in Java with Object Oriented Constructs and planned with Systems Analysis & Design.

For this project we used the Netbeans IDE and implemented a User Graphic Interface using Java Internal Frame.

For the realization of the graphs in UML we used the Visual Paradigm program, the UML graphs made were Class Diagram, Sequence Diagram and Use Case Diagram.

The Diagrams were made in order to help in visualizing and implementing the codes made in Java, so that it would help us with how the codes and classes would look, in addition to helping us to visualize future problems that we would face for the implementation of the software.

Problems Definition

Client Xtra-vision Xpress

Problems with current system

- It's difficult to have and maintain many stores around Dublin;
- The stores takes more time to process all the renting and returning;
- The stores take to much space in the stock to allocate all the movies;
- Staff have a feeling that customers are not very happy to deal with the regular system. It takes a lot of time and sometimes they are upset with customer services.
- The renting and payment process takes longer specially when is a new customer

Objectives of new system

- To provide an efficient and faster hire and return process;
- Simplify the situation about renting and payments process specially when is a new customer;
- To improve customer satisfaction;
- Have more rentability with the quioskies;
- Get more new customers.

Scope of new system

- The whole project will be developed to have more productivity and quicks to rent and return films in kiosks:
- Rent procedures;
- Return procedures;
- Recording new customers;
- Speed and efficiency;

List of system requirements

The XtraVision system must:

R1 - Keep a complete list of movies that they have available to rent. Including the amount of movies available, and details about the movie.
R2 - Keep the records about customers such as credit card name, number and if required by the customer the email also.
R3 - Work automatically the price about each movie, and the total about all movies the customer wants to rent.
R4 - Record all the details about the day and time the movies were rented and returned.
R5 - In case the customer doesn't return the movie in time or not at all the credit card that is in the database will be used to pay the fee.
R6 - The customer will be provided with the order number to return the rented movies.
R7 - If the client is new he is not allowed to rent more than 2 movies.
R8 - All customer information will be kept safe and confidential in our database.

System life cycle

Requirements

Customers are complaining about the time it takes to rent and return a movie, and the fact that there aren't many stores around Dublin.
We will Make a software to introduce kiosks and install these kiosks around Dublin.

Analysis

The system is simple and easy to use. The system will be easy to use from the customer's perspective, will be simple, and no login needed. And will be a simple system delivering the DVD to the customer. In case the customer prefers the receipt will be delivered by email.

Design

The system will be based on how xtravision works, renting movies through a kiosk with a simple system and without a login. All resources on the implementation of the system will be through the XtraVision kiosks.

Implementation

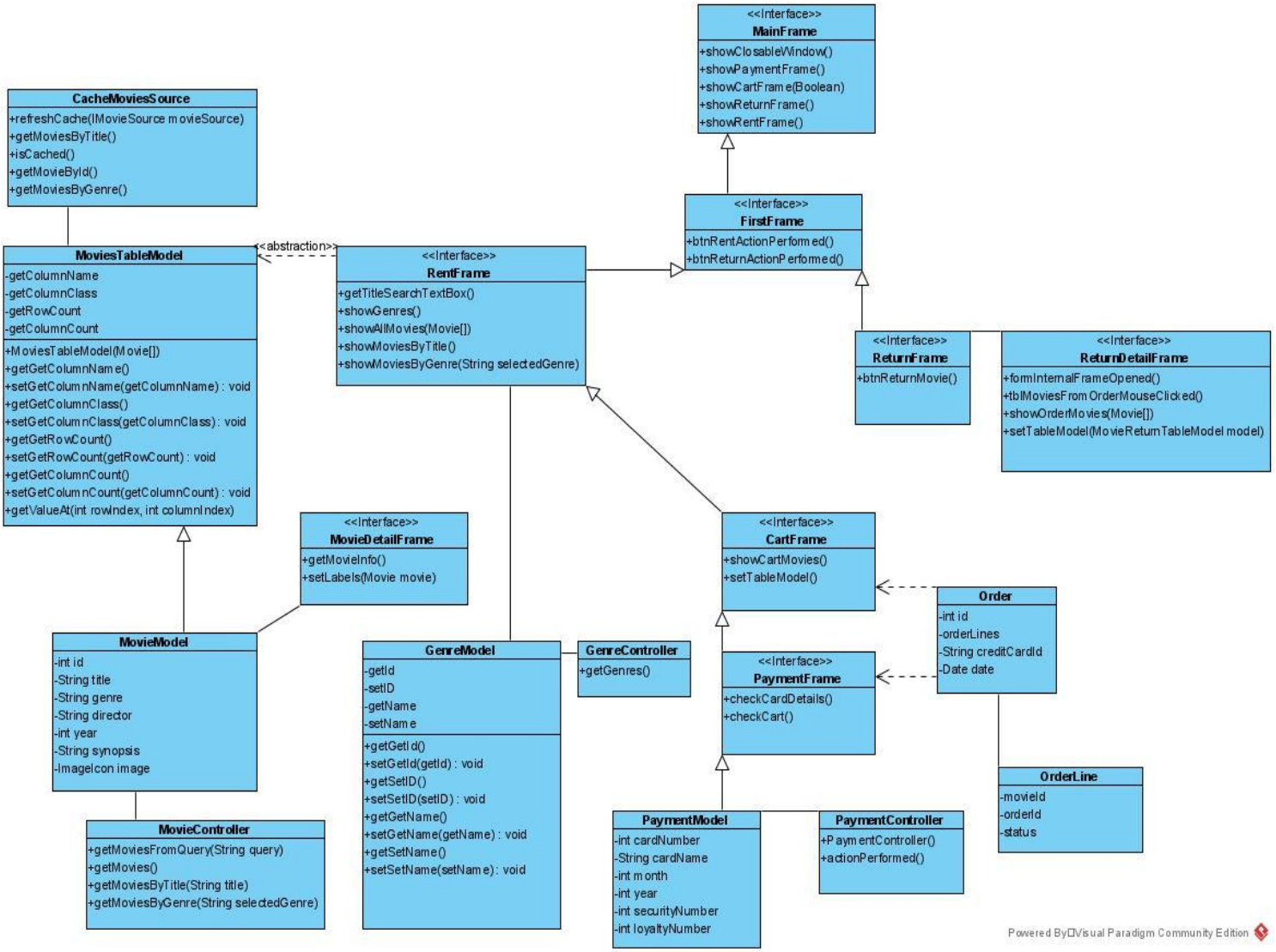
All the diagrams were made to have a base of how we will create each class, and its attributes, improving when making the codes if necessary. We are testing the program to check if there is any bug or improvement.

Installation

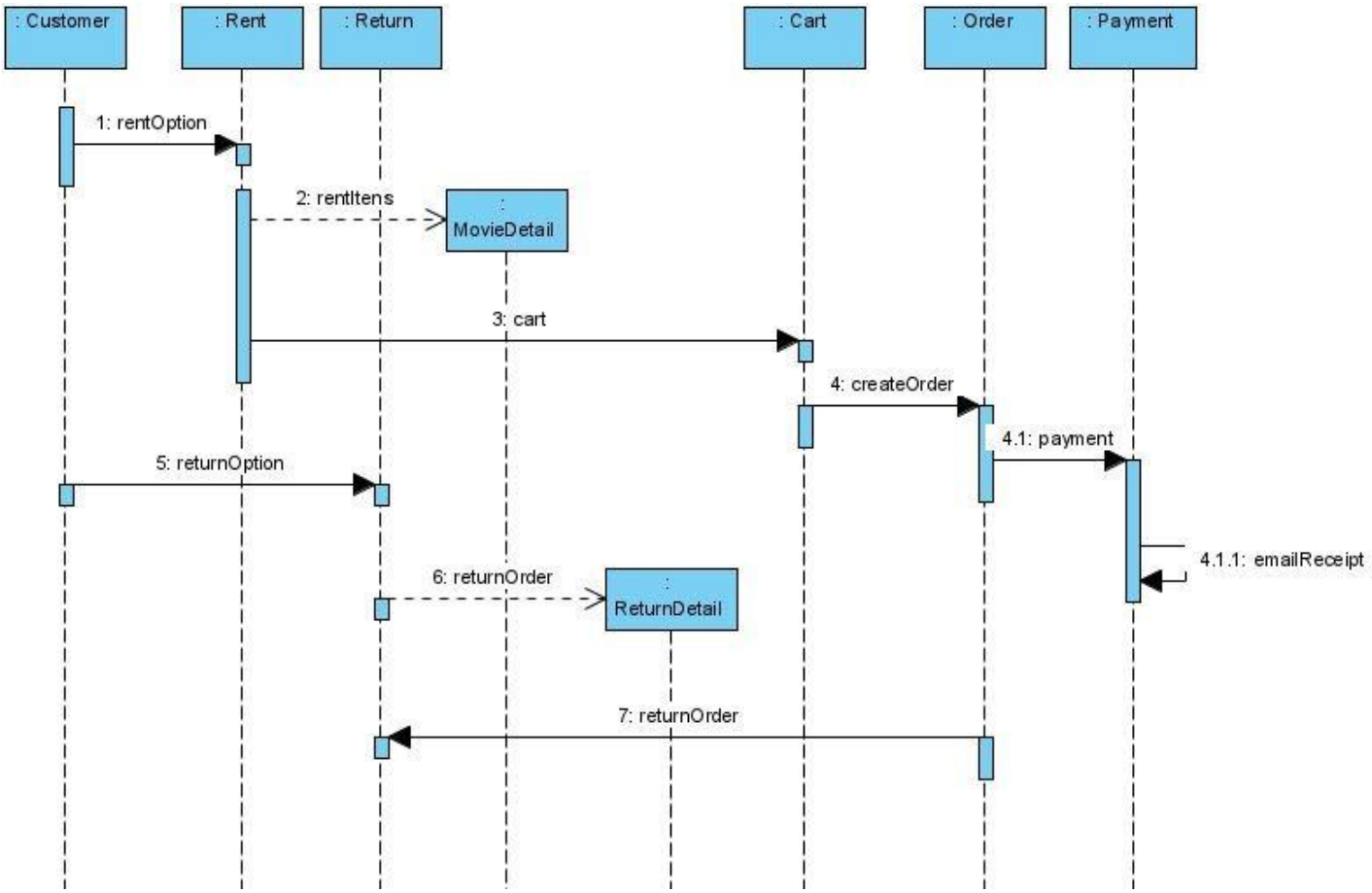
We will gradually implement the program in some kiosks close to the stores, so we will have a staff to collect information and feedback about the system. After that we will implement it in all kiosks.

UML Diagrams

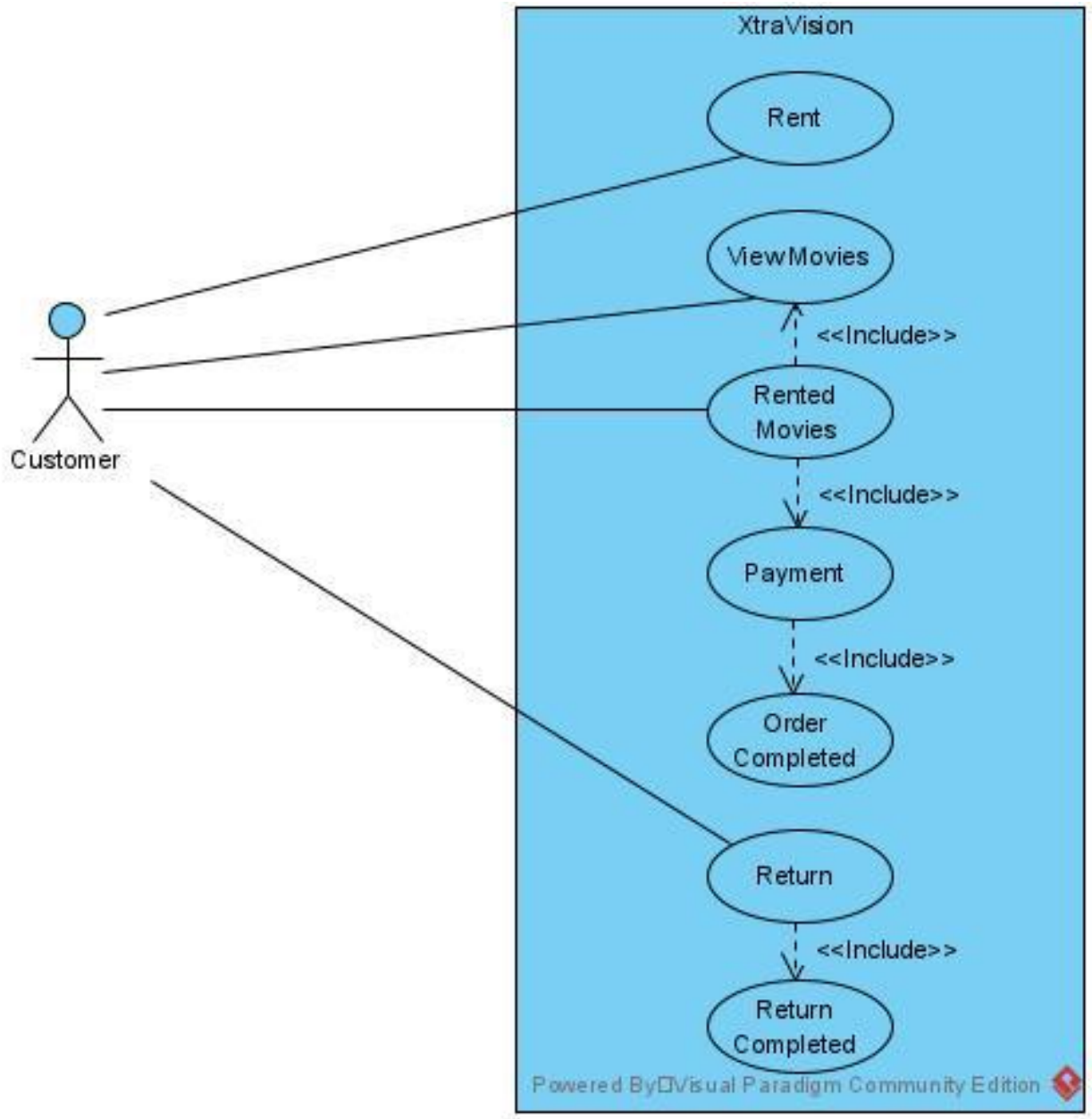
Class Diagram



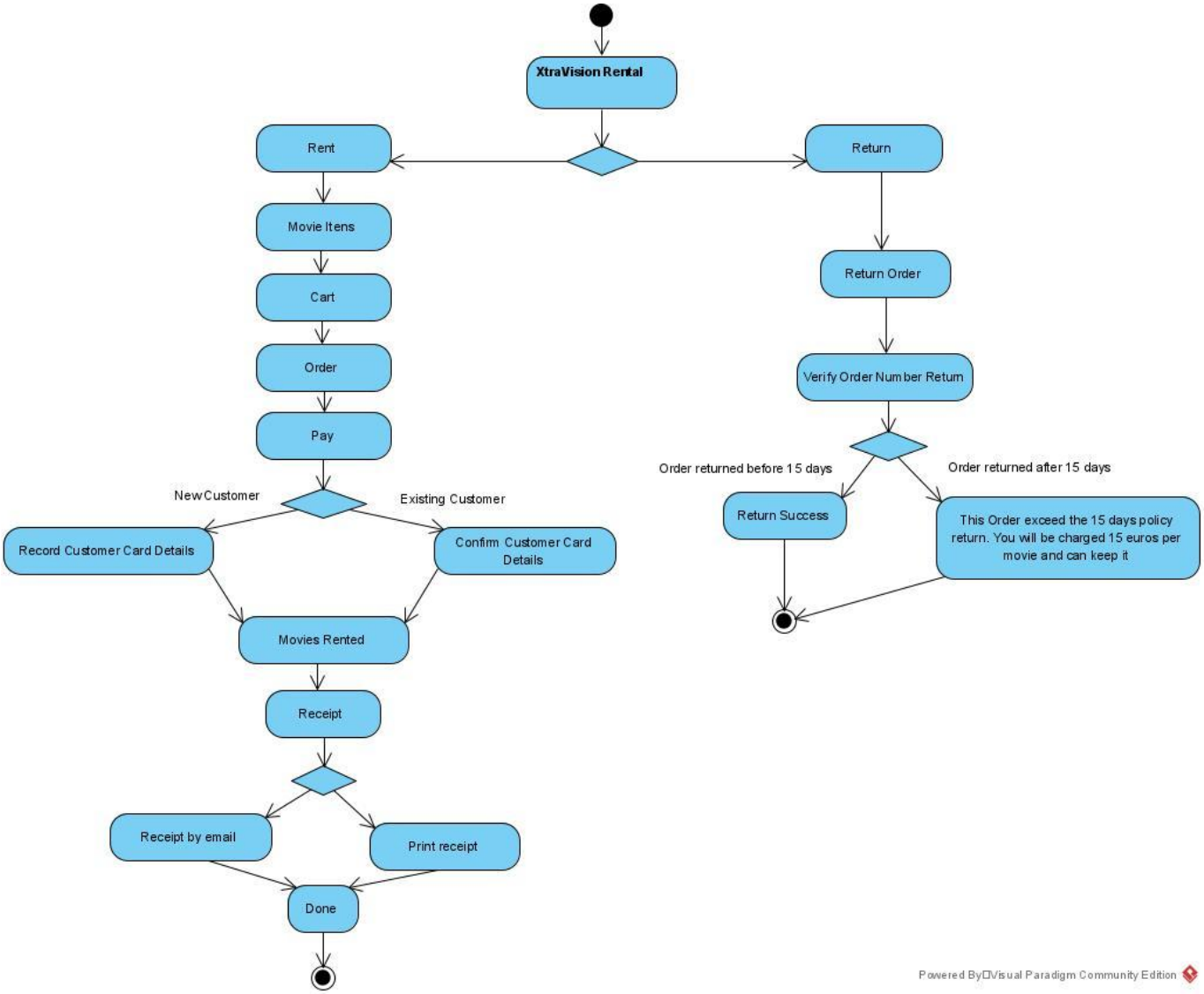
Sequence Diagram



Use Case Diagram



Activity Diagram



Research Software Testing

Software testing is usually the last step in building a program, with a view to checking its quality level. The defects that a test seeks to identify include compatibility error, some algorithm, requirements that cannot be supplemented, and hardware limitation. The bigger the program, the more the list of requirements increases.

Types of software tests

There are different types of tests that can be applied to software to identify its flaws.

The white box test uses the internal aspect of the program or system, the source code, to evaluate its components. It is also known as a logic-oriented or structural test. Items such as: data flow, condition, cycles can be analyzed. When implementing it, it is necessary to check the criticality, complexity, structure and level of quality that is intended to be obtained from the program, involving trust and security.

The black box test is different from the previous test, which prioritizes internal aspects, the black box test checks external aspects. The functional requirements of the system are assessed. The mode of operation, its operation is not observed, focusing on the functions that should be performed by the program. Thus, it is evaluated whether a data entry group resulted in the intended outputs, taking into account the program specification. That is, what the software was expected to do. It is also known as a functional technique;

The gray box test joins the two previous ones, hence the term “gray”. It assesses both internal and external aspects, both inbound and outbound. Reverse engineering can be used.

Regression testing, this consists of performing tests on each version of a software, where functionality is modified. In this way, it avoids errors that were corrected before in the software before appearing again when adding something new to it.

Unit testing, you test smaller units of software, in isolation, to see if they all work properly;

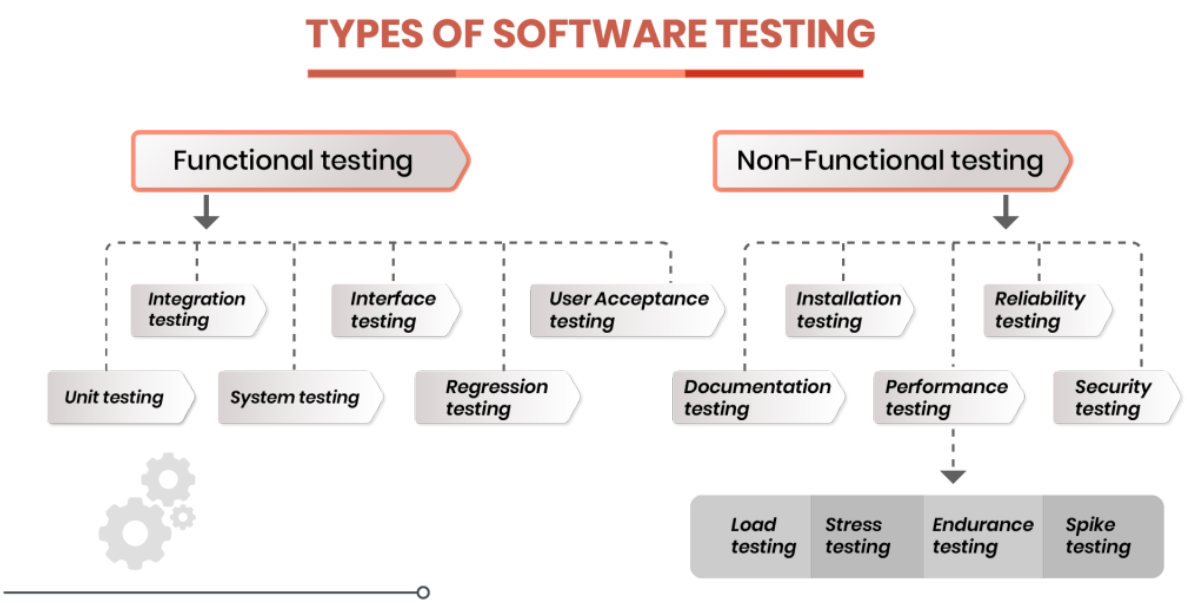
Integration test, after the units tested, a check is made if they work together, integrated. It may occur that they have incompatibilities when working together, even after having passed the unit test;

Load test, this test is done to evaluate the limits of use of the software, how much it supports in volume of information, traffic etc. without presenting errors;

Usability test, this test is done by a small group of users to see if the software meets their needs. In this test it is analyzed how the user uses the system, checking where he has the most difficulty. Their impressions are also heard, but it is necessary to confront them with the evaluator's observations;

Stress test, here the software is taken to its limit of power and functioning, more or less, in order to evaluate at which point it stops working properly.

Figure 1 - Types of Software Testing

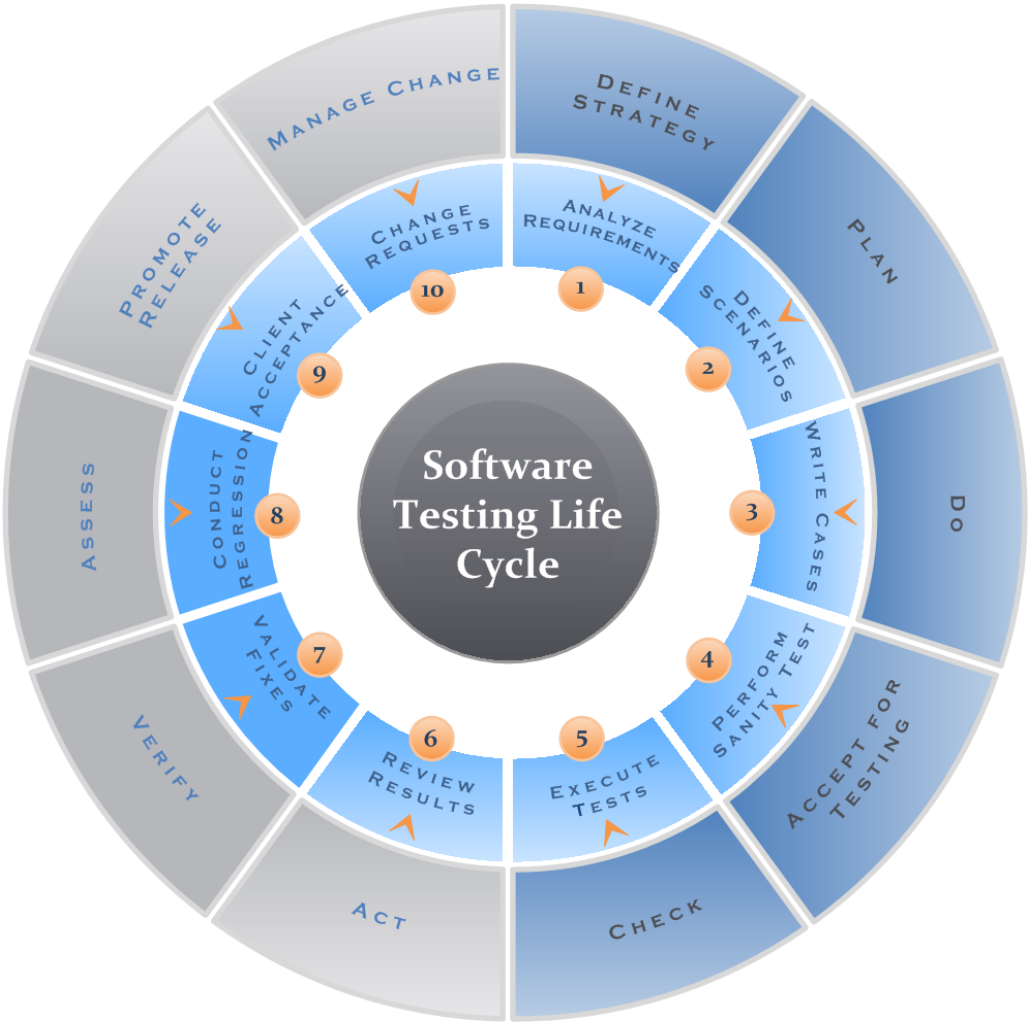


Source: ehikioya.com

A test plan is made to collaborate with software development. It is through this plan that the technical, functional, and structural components will be checked and validated, in order to guarantee the smooth functioning of the program with the end user. Therefore, a software test plan focuses on ensuring the reliability and security of a software, identifying possible errors and flaws during its manufacture, or even afterwards.

It must be planned together with the software proposal, being applied at each stage of the project and not only at the end.

Figure 2 - Software Testing Life Cycle



Source: reqtest.com

Software Testing Process

The Software Testing Process represents a structuring of stages, activities, artifacts, roles and responsibilities that seek to standardize the work and expand the organization and control of the test projects.

The Test Process, like any other process, must be continuously reviewed, in order to expand its performance and enable professionals to have greater visibility and organization of their work, which results in greater agility and operational control of the test projects.

Test Planning is the stage that is characterized by the definition of a testing proposal based on the Client's expectations regarding deadlines, costs and expected quality, making it possible to dimension the team and establish an effort according to the needs pointed out by the Client.

Dynamics of Macro-Activities is the diagram that represents the sequence of the “macro-activities” to be performed in the “Test Planning” stage.
Detailing of Macro-Activities is a list that represents the set of activities that must be performed so that each macro-activity is considered finalized, functioning as a "check-list" for the execution of the "Test Planning" stage.

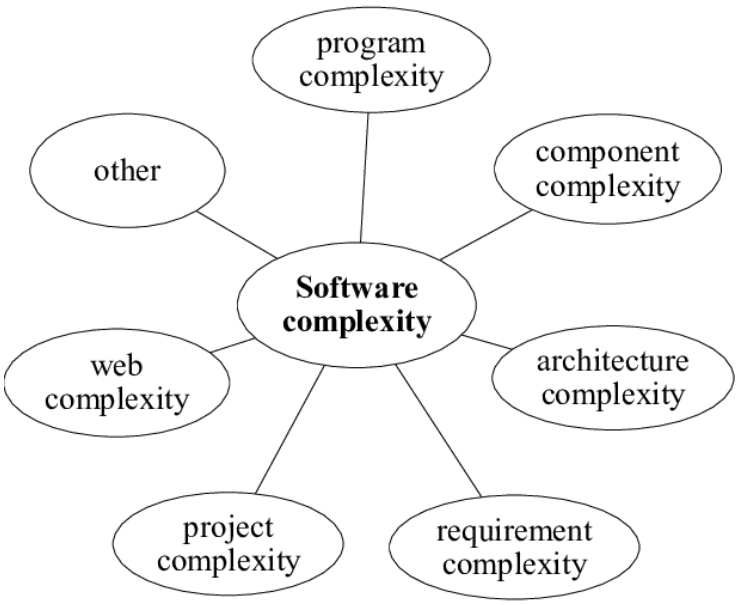
Research regarding dealing with complexity

Software developers need to find solutions to overcome complexity and obstacles and meet the demands of an increasingly demanding audience, including large corporations. The more elements a system or software has, the more relationships and decisions the set has, the more complex it is to understand and validate. A system is a set of elements coexisting with the purpose of reaching a certain measurable objective - possible to be perceived and validated.

In no way is there a single way to design systems, neither the most correct. When it comes to software, there is almost never a canonical solution.
A good strategy is to understand and architect the system as a set of single responsibility subsystems exchanging information with each other.
The smaller the system and with clear relationships between its elements, the faster it can understand the objective and what to expect to deliver. Consequently, identifying defects faster and implementing improvements more often.

Often the systems begin to be modeled with low complexity, but they become highly complex over time. It may have started by modeling only a small part of the process, which, if successful, may require the system to grow.

Figure 3 - Software Complexity



Source: researchgate.net

The quality and robustness of the software are important to maintain the quality of the software and avoid unnecessary complexities.
The more quality and robustness the software presents, the less prone to failure and the less dependent on maintenance and replacement it will be. Customers do not want to have high maintenance costs and the need to replace the product after a short time of use. For customers, this is a poor investment.

Above all, the software must have automatic updates and be easy to use. Thus, the customer will not have great difficulties in using it, nor will he run the risk of compromising it due to improper use.

Another factor that influences the agility of the processes is the use of the most up-to-date technology. It is also advisable to measure, using automated tools, the performance indices of the teams, resources and software itself.

Bibliography

Guru99.com. 2021. What is Software Testing? Definition, Basics & Types. [online] Available at: <https://www.guru99.com/software-testing-introduction-importance.html> [Accessed 8 April 2021].

Atlassian. 2021. The different types of testing in Software | Atlassian. [online] Available at: <https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing> [Accessed 8 April 2021].

TestLodge Blog. 2021. What is a Test Plan in Software Testing? - TestLodge blog. [online] Available at: <https://blog.testlodge.com/what-is-a-test-plan-in-software-testing/> [Accessed 8 April 2021].

Reqtest.com. 2021. [online] Available at: <https://reqtest.com/wp-content/uploads/2016/05/stlc1.png> [Accessed 8 April 2021].

AltexSoft. 2021. 11 Ways to Improve Software Testing through Planning, Work Environment, Automated Testing, and Reporting. [online] Available at: <https://www.altexsoft.com/blog/engineering/software-testing-qa-best-practices/> [Accessed 8 April 2021].

Ehi Kioya. 2021. Understanding The Different Types Of Software Testing - Ehi Kioya. [online] Available at: <https://ehikioya.com/forums/topic/understanding-the-different-types-of-software-testing/> [Accessed 8 April 2021].

Ehikioya.azureedge.net. 2021. [online] Available at: <https://ehikioya.azureedge.net/wp-content/uploads/hm_bbpui/86828/fh2aspgg9au0icjnuf8d9lvhwccrv977.png> [Accessed 8 April 2021].

Jiang, Rong. (2015). Research and Measurement of Software Complexity Based on Wuli, Shili, Renli (WSR) and Information Entropy. Entropy. 17. 2094-2116. 10.3390/e17042094. [Accessed 8 April 2021].

Medium. 2021. Software: Managing the Complexity. [online] Available at: <https://medium.com/@gaperton/software-managing-the-complexity-caff5c4964cf> [Accessed 8 April 2021].

Atlassian. 2021. The Top 8 Tips for Managing Complex Software Projects. [online] Available at: <https://www.atlassian.com/team-playbook/plays/ludicrously-complex-software-projects> [Accessed 8 April 2021].