

Video Games Store

Engineering Method

Jhan Carlos Carvajal Bastidas

Gilmar Andrés Amézquita

Faculty of Engineering, ICESI university

Algorithms and Data Structures

Anibal Sosa

September 2021



Engineering Method

Phase 1: Problem identification.

**Identification of necessities:**

* Know which are the video games that most interest store customers.
* Users of the software system need to know the availability of video games and the location on each of the shelves.
* Users require the fastest route to collect the games in the shortest time possible.
* Checkout cashier queue must be efficient.

**Definition of the problem:**

The store owner requires the development of a software system that allows users to find the games they want to buy in the fastest way and that are efficiently served at the cashiers. In addition to the fact that the shop’s catalog must be always up to date.

Phase 2: Information gathering:

**Database**: a database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is used when large amounts of data are handled and accessed by many users at the same time.

**Spreadsheet**: a spreadsheet is useful if the information it store is not very complex and if the number of users who handle it are minimal.

**Fastest route**: It refers to the order in which the client must collect the games to consume the minimum possible time. If the store is going to have a single aisle with shelves, the order is to go from shelf to shelf, starting at the one closest to section 2.

**Queue at the cashier:** It is the queue that a cashier has for customers to pay for products. Usually there are several queues, but is not the most efficient way, a single queue for multiple cashiers is best.

Phase 3: Searching for creative solutions:

**Alternative 1:**

Design software for different types of users who are going to access and interact with the game store's database. Customers will only be allowed to access the list of games and create their own list to buy. The people in charge of the cashiers in the store could only modify the number of games and register purchases. Store managers would be allowed to modify the catalog.

**Alternative 2:**

Design a software prototype in which the system will interact transversally with the appropriate users in each of the phases, for which the software moments must be separated between the different phases that will be in the store.

Phase 4: From ideas to preliminary designs:

**Alternative 1:**

* This design implies that there must be many users managing the store data at the same time.
* Implement an account system to take care of the information.
* Must implement a database to manage the information of the store.
* Less control between the phases the clients are in the store.

**Alternative 2:**

* This design implies a smaller number of users.
* The software needs to limit the actions of the user depending on the phase where it is.
* A spreadsheet can be implemented to manage the data of the store, because this not have much information and have smaller users.

Phase 5: Evaluation and selection:

The maintenance of a database would consume a lot of resources, which would be unnecessary for the store in case of using a spreadsheet. Also, the store is not going to handle a lot of complex data, so a database is not necessary. However, if it becomes necessary, data migration is a viable option.

So alternative 2 is better this time.

Phase 6: Preparation of reports and specifications:

*Problem specification:*

Problem: customers’ exit order.

Input: list of clients with the games to buy.

Output: list of clients in exit order, with the games they could buy.

*Considerations:*

1. The number of units available for each game must be considered in the selling.
2. The number of clients equals the number of game code lists.
3. There must be at least 1 cashier available.

Phase 7: Design implementation:

List of tasks to implement:

1. Receive game catalog.
2. Receive the list of clients and the games they will buy.
3. View the list of clients’ statuses after phase 2.
4. View the list of clients’ statuses after phase 3.
5. Calculate the games value.
6. View the order of payment and exit.

|  |  |
| --- | --- |
| Name: | importCatalog |
| Description: | Receive the games catalog of the store. |
| Input: | * a: File, it has the catalog saved. |
| Output: | Void. |

|  |  |
| --- | --- |
| Name: | process |
| Description: | Receive the quantity of clients and they game list to buy. Also, some information of the store. |
| Input: | * a: int, quantity of cashiers. * b: int, quantity of clients. * c: String, games’ code. |
| Output: | Client [], list of clients with the games. |

|  |  |
| --- | --- |
| Name: | clientsAfterPhase2 |
| Description: | Show the clients’ statuses (order) at the end of phase 2. |
| Input: | * a: Clients [], list of clients with the games. * b: int, quantity of clients. |
| Output: | Clients [], updated the order of clients’ statuses (order). |

|  |  |
| --- | --- |
| Name: | clientsAfterPhase3 |
| Description: | Show the clients’ statuses (order) at the end of phase 3. |
| Input: | * a: Clients [], list of clients with the games. * b: int [], quantity of games per client. |
| Output: | Clients [], updated the order of clients’ statuses (order) |

|  |  |
| --- | --- |
| Name: | gamesPrice |
| Description: | Calculate the price of the games that each client bought. |
| Input: | * a: Clients [], list of clients with the games. * b: VideoGames [], list of video games with their prices. |
| Output: | Int [], list of prices for each client. |

|  |  |
| --- | --- |
| Name: | exitOrder |
| Description: | Shows the clients’ order of exit. |
| Input: | * a: int, cashiers quantity. * b: Clients [], list of clients. |
| Output: | String, string with the client information, price of games and the list of games’ code. |