Domain Model and System Sequence Diagram

William Carlstedt

March 2023

Introduction

The success of a retail store largely depends on its ability to provide a seamless and efficient shopping experience to its customers. To achieve this goal, it is essential to have a well-designed system that can handle the sales process effectively. This report presents a domain model and system sequence diagram for a retail store, which aims to provide a clear understanding of the structure of the program that we will be building later. The domain model depicts the entities, attributes, and relationships between them, while the system sequence diagram illustrates the basic and alternative flows of the Process Sale. Both diagrams serve as a visual representation of the requirements specification and business rules, providing a foundation for the development of the program. By analyzing the diagrams, we can gain insights into the sales process and identify potential areas for improvement. Models such as these serves as a valuable resource for the development team and stakeholders, helping to ensure that the final program meets the needs of the business and its customers.

Method

Domain Model

1. Use Noun Identification to Find Class Candidates
2. Use a Category List to Find More Class Candidates
3. Remove Redundant Class Candidates
4. Decide Which Classes Fit Better as Attributes
5. Add Associations
6. Check for Mistakes
7. Rework/Rewrite

System Sequence Diagram

The process of creating a system sequence diagram can be approached in a straightforward manner. The first step involves taking the flow specification and carefully examining each point. From there, the relevant information can be transcribed into UML notation, ensuring that the diagram accurately represents the sequence of events described in the specification.

Result

Domain Model

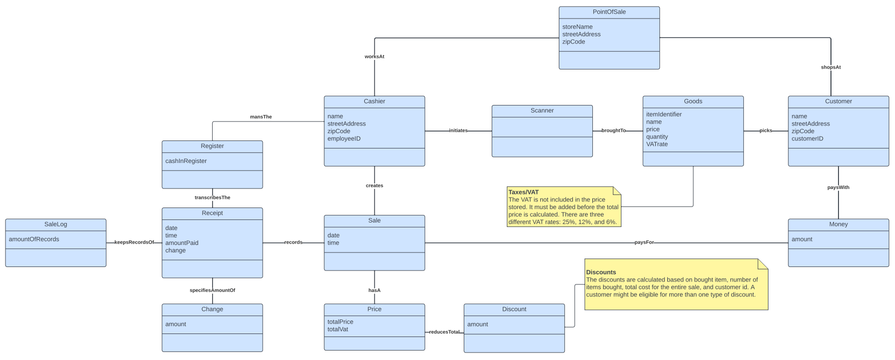

Figure 1: Domain Model

System Sequence Diagram

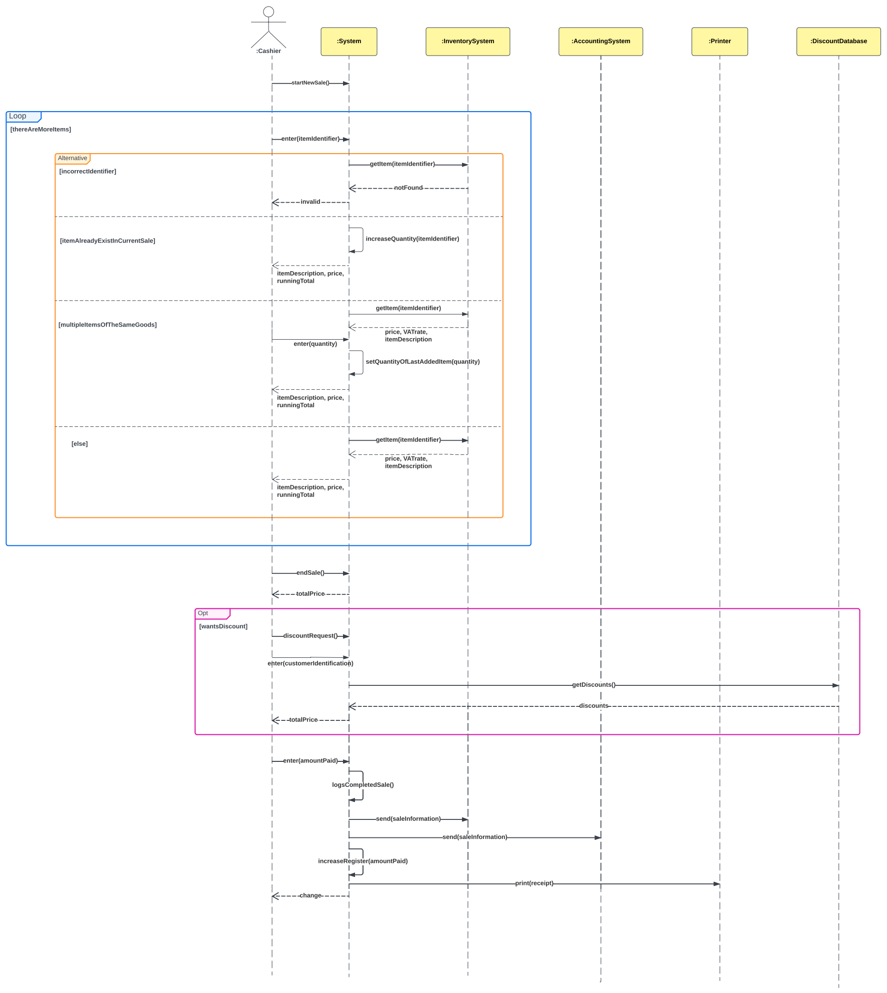

Figure 2: System Sequence Diagram

Discussion

Domain Model

The Domain Model has been developed with the intention of representing the reality, rather than serving as a diagram of the program. While it uses programming naming conventions such as PascalCase for class naming and CamelCase for attribute and association naming, the model is not intended to reflect the implementation details of the program.

However, there are certain concerns and details in the current domain model that require consideration for potential improvement. Firstly, some may perceive the **SaleLog**, **Sale**, and **Cashier** classes' possession of four associations as a ”spider-in-the-web” class structure. While this structure usually refers to a single class with an excessive amount of associations, it may still be argued that the current associations hinder the model's overall flow.

Secondly, it may be more efficient to establish a direct association between **Cashier** and Goods without involving the **Scanner** class, potentially rendering the **Scanner** class redundant. It is also noteworthy that despite the close relationship between **Goods** and **Sale**, there is currently no association between the two classes. However, the addition of such an association would further increase the **Sale** class's already four associations.

System Sequence Diagram

The System Sequence Diagram appears to have some rendering issues, with some lines appearing thicker than others, which is not intentional. This discrepancy is not present in the original Lucidchart version, and it is unclear why it is occurring in the PDF format.

As for the content of the diagram, there is not much room for creativity, as the purpose of the System Sequence Diagram is to clearly outline the steps involved in a system's operation. The correct arrows have been utilized to indicate the calling of operations and returning of values. Any issues with the diagram are more likely to be found in the domain model, which offers greater flexibility in its creation.