Software Engineering Approach For Designing Retail Information Systems

Youssef Bassil

LACSC – Lebanese Association for Computational Sciences
Registered under No. 957, 2011, Beirut, Lebanon
youssef.bassil@lacsc.org

Abstract— Software Engineering is an engineering discipline that is concerned with all aspects of software production including both scientific and technological knowledge, methods, design, implementation, testing, and documentation of software. Today, a successful software project is initially designed by individuals who follow well-defined engineering approaches to problem-solving. This paper discusses the design of an information system related to computer retail store using thorough and strict software engineering practices and principles. The Waterfall software development life cycle model is exploited to analyze, design, and build the proposed system through following a development process made up of a list of phases that must be executed in sequential order. Furthermore, the functional and non-functional requirements, in addition to the business rules, project scheduling and planning, and design specifications are to be discussed. In the design specifications, several detailed illustrations are presented, they include drawings, diagrams, and schemas including but not limited to Viewpoints, Context Models, Use-Cases, Data Flows, and User/System Interactions. As future work, the implementation phase is to be tackled while showing how the design specifications can be transformed into a tangible software, database, and components through writing algorithms, coding, and initial deployment.

Index Terms— Software Engineering, Information Systems, Retails Systems, System Analysis, Waterfall Model, SDLC, Project Management

1. Introduction

When the first digital computer was invented in the early 1940s [1], the instructions that control it and make it operate were hardwired into the computer's hardware. Later, practitioners realized that they needed a way to separate the hardware from the software so as to make it easier working with computers [2]. The Von Neumann architecture [3] was then proposed which makes distinct division between computer hardware and software, and allows instructions to be stored in computer's memory rather than being manually wired into the hardware [4]. From this very moment forward, the revolution in software design and engineering took a different direction. Fundamentally, Software Engineering is an engineering discipline that is concerned with all aspects of software production including both scientific and technological knowledge, methods, design, implementation, testing, and documentation of software [5]. Software engineering can be divided into sub-disciplines including software requirements, software design, software construction, software testing, and software maintenance [6]. Today, the majority of big information systems are oriented toward retail and sales industries. They mostly exhibit a combination of a relational database, an easy to use Graphical User Interface, and software logic. Those systems are initially designed by individuals who follow well-defined engineering approaches to problem-solving. They include engineering models, engineering processes, engineering project management, engineering requirements, and engineering tools.

This paper discusses the design of an information system related to computer retail store. Its major task is to manage and automate the different organization business activities including DVD rentals and sales, software and game sales, and hardware installation and maintenance. Besides, a thorough and strict software engineering practices and principles will be used to build the various components of the proposed system.

2. Proposed Solution

This paper discusses the design of a complete software information system that reliably and efficiently automates the whole business activities of a computer retail store. It defines operational processes, procedures, interfaces, data flows, use-cases, and computer-user interactions for employees, customers, and business owners. The system is mainly composed of a database where all critical data and information are stored. Furthermore, the system features an intuitive Graphical User Interface that manages the logic and functionalities of the system including generating reports, conducting queries, updating, inserting, and deleting data records. The proposed information system can be thought as being a Socio-Technical system [7] as it includes a data repository, a basic interface and complete orientations and procedures on how to use and interact with the system. Moreover, the system can be thought as being a Critical System as a system failure might result in an economic loss and damages to the business. Therefore, being socio-technical and critical at the same time, the proposed system must be carefully designed so as to tolerate user errors and mistakes, handle all exceptions, endure internal faults, and provide robust functionalities.

3. System Development Life Cycle

The System Development Life Cycle or SDLC is a process for software and system development. It comprises several stages to execute in sequence in order to plan, design, build, implement, and maintain an information system. This paper employs the Waterfall model [8] which is one of many types of SDLCs. Basically, the Waterfall model consists of multiple stages that must be executed one after the other and transiting to the next stage only when its preceding stage is completed. Figure 1 shows the different stages of the Waterfall model.

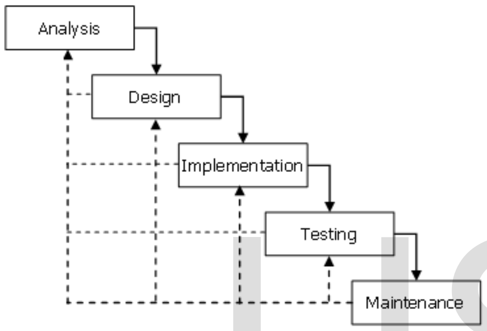

Figure 1: The Waterfall Model

Fundamentally, the Waterfall model consists of five stages: Analysis, design, implementation, testing, and maintenance:

- *Analysis Stage:* In this stage, system requirements are gathered and collected from users, decision makers, and business owners. They comprise both functional and non-functional requirements. The functional requirements are those related to the logic and actual operational functions of the system such as database requirements, interface requirements, process requirements, and function requirements. On the other hand, the non-functional requirements are those related to constraints and limitations imposed on the system such as system reliability, system scalability, system security, system performance, and system quality standards.
- *Design Stage:* In this stage, the functional and non-functional requirements are turned into blueprints and design specifications. It is the process of planning a concrete solution for the software under development. It includes software architecture design, algorithm design, database conceptual schema, uses-cases, data and process flow models, system interaction models, and logical diagram design.
- *Implementation Stage:* In this stage, the design specifications are converted into real executable programs through coding and programming. It is where the final operational software is written and compiled into a production environment.
- *Testing Stage:* In this stage, verification and validation are performed. It is the process of verifying whether or not the implementation meets the original requirements and specifications of the system [9]. Besides, the testing stage is the venue for performing system debugging where bugs, errors, and system glitches are detected, corrected, and fine-tuned accordingly.
- *Maintenance Stage:* In this stage, the developed system is re-evaluated after delivery and deployment to improve its output, correct its logical errors, and enhance its performance and quality. Additional maintenance activities can be carried out such as adapting system to its new environment, adding new user requirements, and improving system reliability [10].

4. Functional Requirements

The proposed information system should meet and comply with the specifications and requirements already agreed on with the business owner, that is the different business rules and operations that govern the daily organizational activities. Next are set of abstract requirements and business rules that would guide the later development and design of the system.

DVD Rental Business Rules:

- A User should first apply to a membership
  → A new record is added to the **Customer's Catalog**
- A User select & rent a DVD
  → A new record is added to **DVD Rentals** Table
  → Quantity in **DVDMovies Catalog** is updated accordingly
- A User returns the rented DVD
  → DVD is marked as returned in **DVD Rentals** Table
  → Quantity in **DVDMovies Catalog** is updated accordingly

DVD Purchases Business Rules:

- A Supplier should first be recorded in **Supplier's Catalog**
  → If new supplier, a new record is added to **Supplier's Catalog**
- The Administrator validates and accepts the offered/ordered products
  → If the ordered product is defined in the **DVDMovies Catalog**
    → Add record to **DVD Purchases** table
    → Update quantity in **DVDMovies Catalog**
  → If the ordered product is not defined in the **DVDMovies**
    → Add record to **DVD Purchases** table
    → Add record to **DVDMovies Catalog**

Hardware Sales Business Rules:

- A User select & buy a hardware
  → A new record is added to **Hardware Sales** table
  → Quantity in **Hardware Catalog** is updated accordingly

Hardware Purchases Business Rules:

- A Supplier should first be recorded in **Supplier's Catalog**
  - → If new supplier, a new record is added to **Supplier's**
- The Administrator validates and accepts the offered/ordered products
  - → If the purchased product is defined in the **Hardware**
    - → Add record to **Hardware Purchases** table
    - → Update quantity in **Hardware Catalog**
  - → If the purchased product is not defined in the **Hwd Catalog**
    - → Add record to **Hardware Purchases** table
    - → Add record to **Hardware Catalog**

Programs & Games Sales Business Rules:

- A User select & buy a Program or Game CD
  - → A new record is added to **ProgramsGames Sales** table
  - → Quantity in **ProgramsGames Catalog** is updated

Programs & Games Purchases Business Rules:

- A Supplier should first be recorded in **Supplier's Catalog**
  - → If new supplier, a new record is added to **Supplier's**
- The Administrator validates and accepts the offered/ordered products
  - → If the purchased product is identified in the **Programs**
    - → Add record to **ProgramsGames Purchases** table
    - → Update quantity in **ProgramsGames Catalog**
  - → If the purchased product is not identified in the **Programs**
    - → Add record to **ProgramsGames Purchases** table
    - → Add record to **ProgramsGames Catalog**

Following are the system functional requirements, standardized in a tabular format from Table 1 to Table 6, which delineate the requirements' function names, their description, the data input source, the output results, and the complete list of actions.

Table 1: DVD Rentals Requirements

DVD Rentals Requirement	
Function	Renting a DVD
Description	Rent a DVD → Enter DVD ID and Customer's Name
Input	DVD ID and Customer's Name
Source	Graphical User Interface
Output	DVD is marked as Rented
Destination	Personal Computer
Action	• A User should first apply to a membership → A new record is to be added to **Customer's Catalog** • A User select & rent a DVD → A new record is added to **DVD Rentals** Table → Quantity in **DVDMovies Catalog** is updated accordingly • A User returns the rented DVD → DVD is marked as returned in **DVD Rentals** Table → Quantity in **DVDMovies Catalog** is updated accordingly
Requires	DVD ID, Customer's Name
Pre-Condition	Customer must have a record in the Customer's Catalog DVD must be available
Post Condition	DVD is marked as rented
Side Effect	None

Table 2: DVD Purchases Requirements

DVD Purchases	
Function	Purchase a DVD
Description	Purchase a DVD → Enter DVD ID and Supplier's Name
Input	DVD ID and Supplier's Name
Source	Graphical User Interface
Output	A new record is added in the DVD Purchases table
Destination	Personal Computer
Action	• A Supplier should first be recorded in **Supplier's Catalog** → If new supplier a new record is added to **Supplier's Catalog** • The Administrator validate and Accept the offered/ordered products → If the purchased product is identified in the **DVDMovies Catalog** → Add record to **DVD Purchases** table → Update quantity in **DVDMovies Catalog** → If the purchased product is not identified in the **DVDMovies Catalog** → Add record to **DVD Purchases** table → Add record to **DVDMovies Catalog**
Requires	DVD ID, Supplier's Name
Pre-Condition	Supplier must have a record in the Supplier's Catalog
Post Condition	Quantity is updated
Side Effect	None

Table 3: Hardware Sales Requirements

Hardware Sales	
Function	Sell a Hardware
Description	Sell a Hardware → Enter Hardware ID and Customer's Name
Input	Hardware ID and Customer's Name
Source	Graphical User Interface
Output	A new record is added in the Hardware Sales table
Destination	Personal Computer
Action	• A User select & buy a hardware → A new record is added to **Hardware Sales** table → Quantity in **Hardware Catalog** is updated accordingly
Requires	Hardware ID, Customer's Name
Pre-Condition	Customer must have a record in the Customer's Catalog
Post Condition	Quantity is updated
Side Effect	None

Table 4: Hardware Purchases Requirements

Hardware Purchases	
Function	Purchase a Hardware
Description	Purchase a Hardware → Enter Hardware ID and Supplier's Name
Input	Hardware ID and Supplier's Name
Source	Graphical User Interface
Output	A new record is added in the Hardware Purchases table
Destination	Personal Computer
Action	• A Supplier should first be recorded in **Supplier's Catalog** → If new supplier a new record is added to **Supplier's Catalog** • The Administrator validate and Accept the offered/ordered products → If the purchased product is identified in the **Hardware Catalog** → Add record to **Hardware Purchases** table → Update quantity in **Hardware Catalog** → If the purchased product is not identified in the **Hardware Catalog** → Add record to **Hardware Purchases** table → Add record to **Hardware Catalog**
Requires	Hardware ID, Supplier's Name
Pre-Condition	Supplier must have a record in the Supplier's Catalog
Post Condition	Quantity is updated
Side Effect	None

Table 5: Program/Game Sales Requirements

Program/Game Sales	
Function	Sell a Program/Game
Description	Sell a Program/Game → Enter Program/Game ID and Customer's Name
Input	Program/Game ID and Customer's Name
Source	Graphical User Interface
Output	A new record is added in the Program/Game Sales table
Destination	Personal Computer
Action	• A User select & buy a hardware → A new record is added to **Program/Game Sales** table → Quantity in **Program/Game Catalog** is updated accordingly
Requires	Program/Game ID, Customer's Name
Pre-Condition	Customer must have a record in the Customer's Catalog
Post Condition	Quantity is updated
Side Effect	None

Table 6: Program/Game Purchases Requirements

Program/Game Purchases	
Function	Purchase a Program/Game
Description	Purchase a Program/Game → Enter Program/Game ID and Supplier's Name
Input	Program/Game ID and Supplier's Name
Source	Graphical User Interface
Output	A new record is added in the Program/Game Purchases table
Destination	Personal Computer
Action	• A Supplier should first be recorded in **Supplier's Catalog** → If new supplier a new record is added to **Supplier's Catalog** • The Administrator validate and Accept the offered/ordered products → If the purchased product is identified in the **Program/Game Catalog** → Add record to **Program/Game Purchases** table → Update quantity in **Program/Game Catalog** → If the purchased product is not identified in the **Program/Game Catalog** → Add record to **Program/Game Purchases** table → Add record to **Program/Game Catalog**
Requires	Program/Game ID, Supplier's Name
Pre-Condition	Supplier must have a record in the Supplier's Catalog
Post Condition	Quantity is updated
Side Effect	None

5. Non-Functional Requirements

In this section, the non-functional requirements (NFR) [11] and constraints are defined which can be used to judge and ensure the efficient operation of the system, and they are:

- **Availability:** The system has to endure a non-stop 16 hours of operation a day while supporting heavy queries and transaction.
- **Reliability:** The system has to sustain major system failures, errors, and faults while coping with substantial operations and processing. As a reliability metric, ROCOF (Rate of failure occurrence) is the benchmark to test and evaluate the system while in operation.
- **Security:** The system has to support the three pillars of system security, namely confidentiality by means of data encryption and user authentication; integrity by means of cryptographic hashing and checksum; and availability by means of redundancy, failover, and RAID architectures [12].
- **Maintainability & Ability to Evolve:** The system has to support performing a successful repair with speed and ease within a given time without interrupting the operational status of the system. As future requirements, new services and business rules might be required, so the system has to be easily updated and upgraded by add-on modules and hot fixes without interrupting its running operations.

6. Project Scheduling & Planning

Project scheduling is a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what timeframe [13]. A project schedule is a well-structured document whose purpose is to set Milestones and Deliverables to show the dependencies between different activities. Table 7 delineates the deliverables along with their development time and dependencies.

Table 7: Deliverables

Deliverables	Days	Dependencies
T1 (Identifying Business Situation)	8 Days	
T2 (Check Technology & Tools)	6 Days	
T3 (Collecting Requirements)	6 Days	T2(M1)
T4 (Analysis Phase)	7 Days	T3(M2)
T5 (Logical Design)	5 Days	T3(M3)
T6 (Implementation)	10 Days	T5(M4)

Figure 2 depicts the Gantt chart that illustrates the actual project schedule. The chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis.

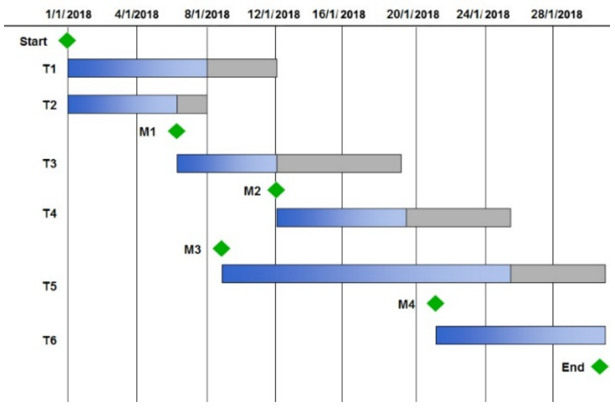

Figure 2: The Gantt Chart

7. Other Constraints

Risk Identification (RI) is an important process that pins out all risks that might arise during system development which would interrupt or even stop the advance in the project processes. Risk is classified as low reasonably practical (ALARP) system whose failure doesn't affect seriously the business. The most obvious risk is the migration of data from old and legacy systems to the new proposed information system. This might be solved by means of data converters and middleware that interface between the two software systems. Other constrains reside in the programming standards in use. As a result a standard CASE tool such as MS Visio is to be used which implements and features all the common modeling standards of the RUP. Furthermore, discussing programming standards, the system is a sort of computer database with integrated GUI and forms implemented under MS SQL Server 2017; while the programming language is C#.NET using the .NET Framework 4.5.

8. Design Specifications

A design specification is a comprehensive document providing information about a designed product or system. It includes all necessary drawings, diagrams, schemas, pseudo-code, and algorithms. Figure 3 depicts the system's Viewpoints which define a comprehensible set of views to be used in the construction of the software system. Figure 4 depicts the Context Models which define the boundary between the system, or part of the system, and its environment, showing the entities that interact with it. Figure 5 depicts the Use-Cases which define actions and events as interactions between an actor and the system itself to achieve a predefined goal. Figure 6 depicts the system's Data Flow diagrams which represent the flow of data of processes inside the information system. Figure 7 depicts the User/System Interactions diagrams which show the real interactions between users, actions, and data inside the system.

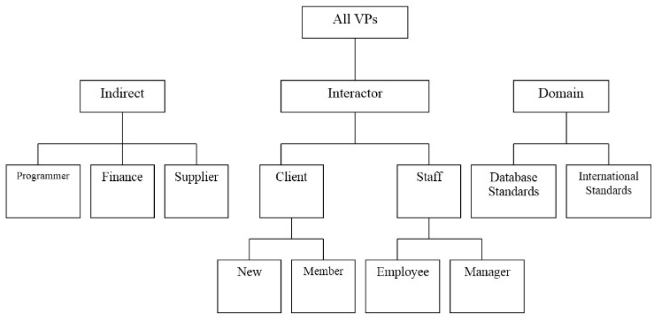

Figure 3: Viewpoints

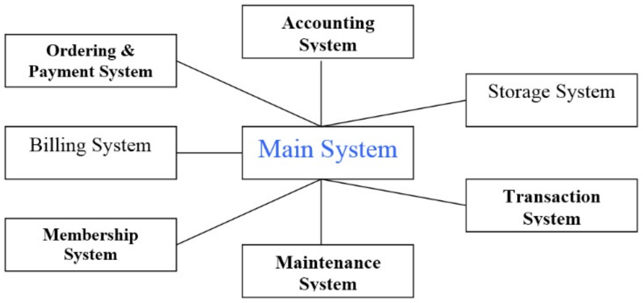

Figure 4: Context Model

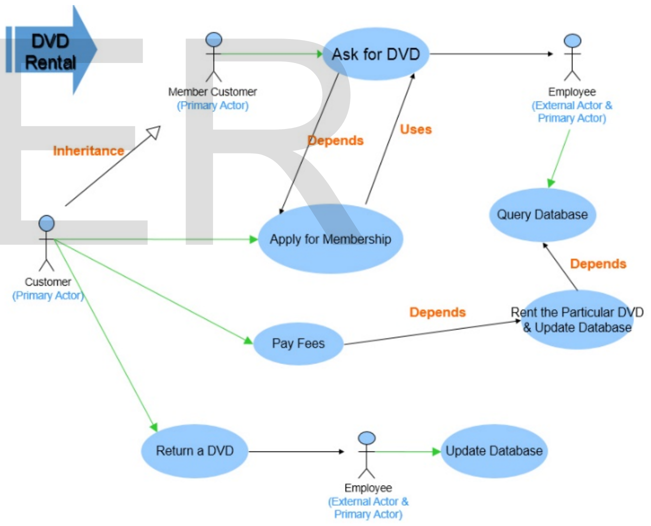

Figure 5 (a): Uses-Cases

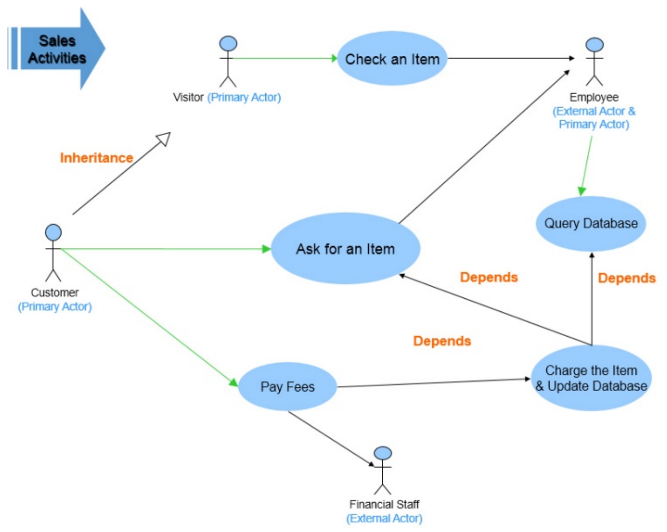

Figure 5 (b): Uses-Cases

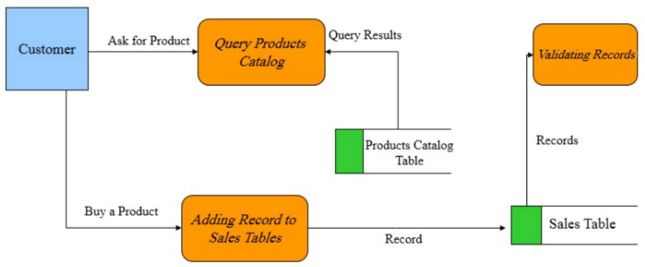

Figure 6 (b): Data Flow

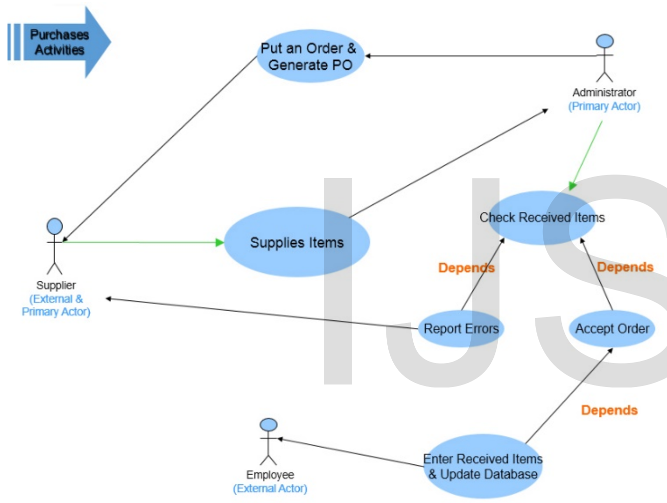
Purchases Activities
Put an Order & Generate PO
Administrator (Primary Actor)
Supplier (External & Primary Actor)
Supplies Items
Check Received Items
Depends
Depends
Report Errors
Accept Order
Depends
Employee (External Actor)
Enter Received Items & Update Database

Figure 5 (c): Uses-Cases

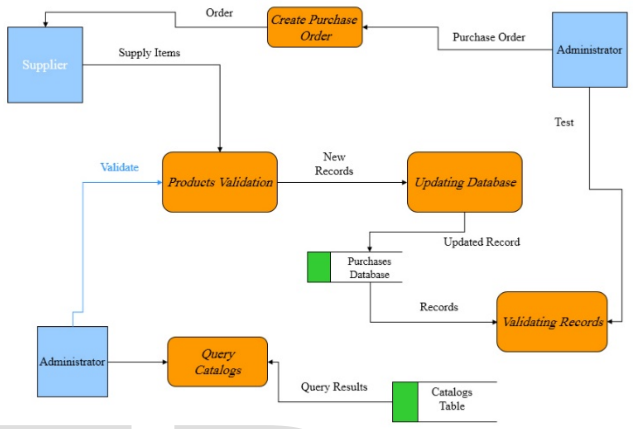

Figure 6 (c): Data Flow

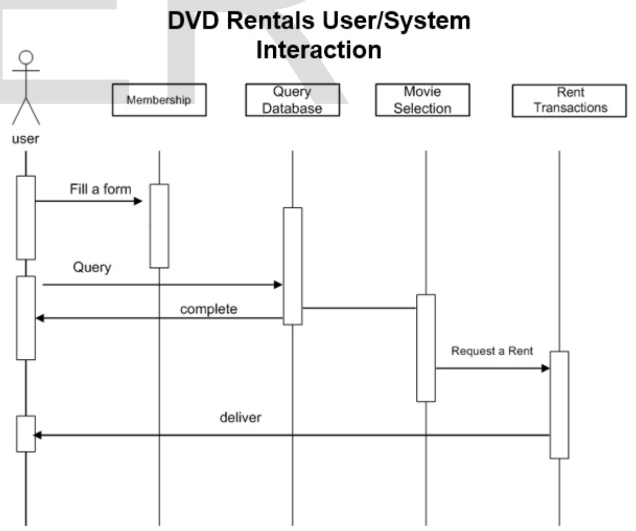

Figure 7 (a): User/System Interactions

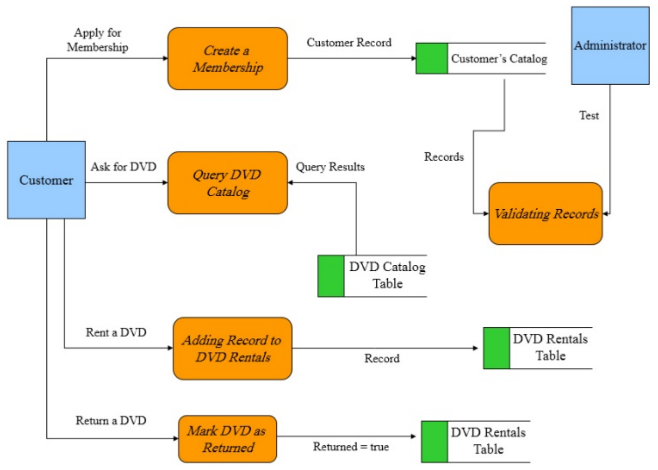

Figure 6 (a): Data Flow

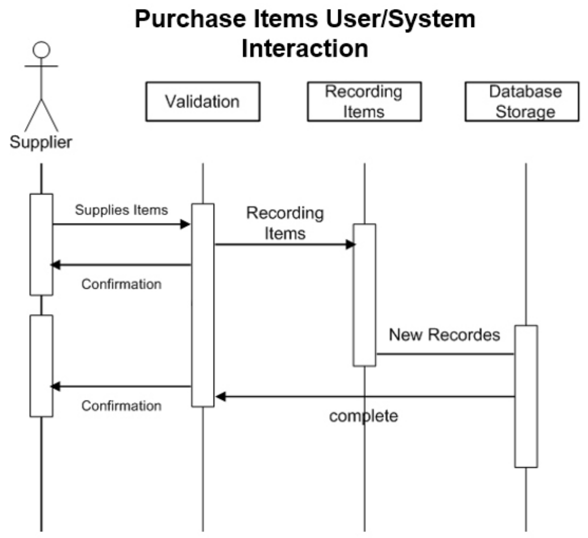

Figure 7 (b): User/System Interactions

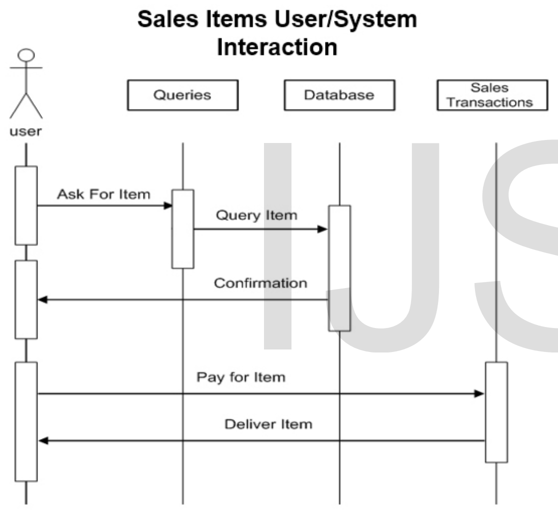

Figure 7 (c): User/System Interactions

9. Conclusions and Future Work

This paper discussed the design of an information system for a computer retail store business. Thorough and comprehensive software engineering practices and principles were used to design the various components of the proposed system. They include the waterfall model as a software development life cycle, functional and non-functional requirements, business rules, project scheduling and planning, and design specifications. Furthermore, several detailed illustrations were presented as part of the software design, they included drawings, diagrams, and schemas including but not limited to Viewpoints, Context Models, Use-Cases, Data Flows, and User/System Interactions.

As future work, the implementation phase is to be tackled showing how the business requirements and design specifications can be turned into a concrete executable program, database, website, and software components through writing algorithms, coding, and initial deployment.

Acknowledgments

This research was funded by the Lebanese Association for Computational Sciences (LACSC), Beirut, Lebanon, under the "Big Data Research Project – BDRP2019".

References

[1] Georges Ifrah, "The Universal History of Computing: From the Abacus to the Quantum Computer", New York: John Wiley & Sons, ISBN 9780471396710, 2001
[2] Arthur Burks, "Electronic Computing Circuits of the ENIAC", Proceedings of the I.R.E., vol. 35, no. 8, pp. 756–767, 1947
[3] John L. Hennessy, David A. Patterson, David Goldberg, "Computer architecture: a quantitative approach", Morgan Kaufmann, ISBN 9781558607248, 2003
[4] Campbell-Kelly, Martin, "The Development of Computer Programming in Britain (1945 to 1955)", IEEE Annals of the History of Computing, vol. 4, no. 2, pp. 121–139, 1982
[5] Sommerville, Ian, "Software Engineering", (8th ed.), Pearson Education, ISBN 9780321313799, 2007.
[6] David Parnas, "On the Criteria To Be Used in Decomposing Systems into Modules". Communications of the ACM, vol. 15 no. 12, pp.1053–1058, 1972
[7] Mills, Harlan D., J. R. Newman, and C. B. Engle, Jr., "An Undergraduate Curriculum in Software Engineering,", Software Engineering Education: SEI Conference, 1990.
[8] Royce, W., "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON 26, pp.1-9, 1970.
[9] IEEE-STD-610, "A Compilation of IEEE Standard Computer Glossaries", IEEE Standard Computer Dictionary, 1991.
[10] Andrew Stellman, Jennifer Greene, "Applied Software Project Management", O'Reilly Media, 2005.
[11] Roger S. Pressman, Bruce Maxim, "Software Engineering: A Practitioner's Approach", 8th ed, McGraw-Hill Education, ISBN13: 9780078022128, 2014
[12] Thomas Schlienger, Stephanie Teufel, "Information security culture-from analysis to change", South African Computer Journal, vol. 31, pp. 46–52, 2013
[13] Murray Woolf, "FASTER Construction Projects with CPM Scheduling", 1st ed, McGraw-Hill, ISBN 9780071486606, 2007