CONTENTS OF THE PROJECT REPORT

1) Cover and Title page

2) Index

3) Certificate of Originality

4) Acknowledgement

5) Certificate/Company Certificate

6) Introduction/Aims and Objective

7) System Analysis

7.1 Identification of Need

7.2 Preliminary Investigation

8) Feasibility Study

8.1 Technical Feasibility

8.2 Economic Feasibility

8.3 Operational Feasibility

9) Analysis (Feasibility Study, DFD 0 Level, 1- Level and 2 Level/ER Diagram, and Data structure, Table structure etc)

10) S/W Engineering. Paradigm applied

11) S/W & H/W Requirement Specification

12) System Design

14) Coding

15) Validation Checks

16) Implementation and Maintenance

17) Testing (Testing techniques and Testing strategies)

18) System Security measures

19) Various types of Reports/Modules (Screen shots)

20) Pert Chart/Gantt Chart

21) Future scope of the project

22) Bibliography/References (in IEEE format)

***Introduction***

**“ZuperMarket (On-Line Shopping System / E-commerce)”** is a web-based project which is made for remote-shopping or shopping through Internet. As the technology is being advanced the way of life is changing accordance. Now a day’s we can place the order for anything from our home. There is no need to go the shop of the things we want. The order can be placed online through Internet. The payment, the confirmation of purchasing; we can do everything that we want. Now we can think that how the days have been changed with time. People had to stand in rows to wait there terms to buy a particular thing from a popular shop. But what is happening now a day’s; we can extremely surprise that those things can be available on the door-step in few hours. People had to suffer the rush of the market when they went for shopping. They used to think hundred times to buy anything having the sufficient money for shopping. The problem was the rush; the quarrel at the time of buying the things. But the advancement of technology brought the new way for shopping. The way of shopping was completely changed with the coming of Internet Technology. People have to fill a simple form on the internet to place their order on any popular shop or shopping-mall for the thing they want to buy. Now they can place their order from the home. This project entitled “On-Line Shopping” is an implementation of the above description. It means, it implements the E-shopping or in other word shopping through Internet. It lets the user to place their order online for any article.

***OBJECTIVE***

Today the internet and its boom have created a new economic scenario that not only stresses on the classical concept of the “product” but also on the modern concept of “service”. It is this level of service that dictates whether a commercial venture will succeed or not in the market. To provide a high accessibility of service we will design the online shopping website, so that potential customers need not go to a physical shop to buy products or services. There are several objective of this websites are following given below:

• This site is gives all the information about the e-shopping to provide better service for the customer.

• It provides the facility to the customers who want to shop on-line due to lack of time.

• It’s providing the full details about the product and related information about the product like cost, size etc.

• With the help of it we can save the time and money also.

• It provides multiple payment methods for shopping by the cash, Debit card and credit card also.

• It provides better security and good delivery service to the customer.

***SYSTEM ANALYSIS***

**Problem Definition / Identification of Need:**

The ZuperMarket (On-Line Shopping System / E-commerce) Administrator is the super user and has complete control over all the activities that can be performed. The application notifies the administrator of all shop creation requests, and the administrator can then approve or reject them. The administrator also manages the list of available product categories. The administrator can also view and delete entries in the guestbook.

**Shop Owner:**

Any user can submit a shop creation request through the application. When the request is approved by the Mall Administrator, the requester is notified, and from there on is given the role of Shop Owner. The Shop Owner is responsible for setting up the shop and maintaining it. The job involves managing the sub-categories of the items in the shop. Also, the shop owner can add or remove items from his shop. The Shop Owner can view different reports that give details of the sales and orders specific to his shop. The Shop Owner can also decide to close shop and remove it from the mall.

**Employees:**

Purchase department under a Purchase manager to overlook purchasing activities if warehousing needs arise. Sales department under a Sales manager who will look after the sale of products and services, the most important activity. Accounts department under an Accounts manager to look after the accounting activities of the enterprise.

**Requirement Specification:**

It is quite difficult and time consuming task to find the information as well as maintaining information manually. If all these information are to be kept at a single place it is also not possible in the manual system. Computerized system will upgrade and manage information very easily. As it is a web-based application so it uses Internet technologies and its hardware/software requirement will also be more comprehensive than Desktop application system. Some Network devices will be required like modems, switches, Internet connection. Software required for the system is also different from a normal desktop system. First of all a server software will be mandatory (here Internet Information Server (IIS)). A browser is also needed as a client process on the user side.

**Preliminary Investigation:**

System Analysis is not only time consuming but also a rigorous task. But it is crucial and most important phase of Software development process. Preliminary Investigation is the process of gathering data for requirement analysis. It is more helpful for problem definition and requirement specification.

***FEASIBILITY ANALYSIS***

Whatever we think need not be feasible .It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. They were three key consideration involved in this feasibility analysis each consideration has reviewed to depict how it relates to the system effort.

They are as follows:-

1. Technical feasibility

2. Economic feasibility

3. Operational feasibility

**Technical Feasibility:**

We can strongly says that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization here we are utilizing the resources which are available already.

Technical feasibility can be evaluated only after those phases during which technical issues are resolved- namely after the evaluation and design phase of our life cycle have been completed. Today very little is technically impossible. Consequently technical feasibility looks at what is practical and reasonable. Technical feasibility addresses three major issues:

* Is the proposed technology or solution practical?
* Do we currently possess the necessary technology?
* Do we possess the necessary technical expertise, and is the schedule reasonable?

**IS THE PROPOSED TECHNOLOGY OR SOLUTION PRACTICAL?**

The technology is very practical for the current system. Organization is using .Netsoftware application support with SQL Database Management Software. PlusMicrosoft Site Server.

**DO WE CURRENTLY POSSESS THE NECESSARY TECHNOLOGY?**

The organization will buy all the necessary technology for example the Pc’s, Routers,Blade Servers, WAN devices, SAN etc

DO WE POSSESS THE NECESSARY TECHNICAL EXPERTISE?

We may have the technology but that does not mean that we have the required expertise and skills to implement the technology effectively. We will hire

**Economical Feasibility:**

Development of this application is highly economically feasible .The organization needed not spend much money for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources .Even after the development, the organization will not be in condition to invest more in the organization. Therefore, the system is economically feasible.

The manual efforts involved in maintaining the record and student information, is tremendous. This is so because the volume of information to be handled is tremendous. Maintaining the records of these many student is not easy and the manpower involved is great.

ALTERNATIVES

1. Hire more staff

2. Develop a computerized system using VISUAL BASIC & SQL server as database provider.

3. Develop a computerized system using VISUAL BASIC & ORACLE as the database provider.

4. Develop a computerized system using C++ & maintaining data file i.e. \*.dat for database.

Its Advantages are:-

1. The system will be platform independent and there will be no need is for purchasing other very expensive software. The jdk/jvm is easily and freely available and is compatible with most of the computers. The system can be upgraded and maintained easily to keep in tune with specific needs of the customer. The memory required by this software will not be much, so we won’t have to upgrade the present system (computer).

The last alternatives suggested by the software development team are the cheapest possible one. The software is text based so no formal training will have to be imparted. The compiler is freely available. The only expenditure that needs to be done is money cost of software team.

**Operational Feasibility :**

It is mainly related to human organizational and political aspects. The points to be considered are:

• What changes will be brought with the system?

• What organizational structures are distributed?

• What new skills will be required? Do the existing staff members have these skills?

• If not, can they be trained in due course of time?

Generally project will not be rejected simply because of operational infeasibility but such considerations are likely to critically affect the nature and scope of the eventual recommendations.

For operational feasibility study we appointed a small group of people who are familiar with information system techniques, who understand the parts of the business that are relevant to the project and are skilled in system analysis and design process.

Operational feasibility criteria measure the urgency of the problem (survey and study phases) or the acceptability of the solution. There are two aspects of operational feasibility to be considered:

* Will the solution to the problem work?
* How do the customer and organization feel about the solution?

**WILL THE SOLUTION TO THE PROBLEM WORK**

PIECES framework can be used as a basis for analyzing the urgency of the problem or the effectiveness of a solution. The following is a list of questions that addressed these issues:

* P- Performance. Does the system provide adequate throughput and response time?
* I- Information. Does the system provide end-users and managers with timely, pertinent, accurate and usefully formatted information .
* E- Economy. Does the system offer adequate service level and capacity to reduce the costs of the business or increase the profits of the business?
* C- Control. Does the system offer adequate controls to protect against fraud and embezzlement and to guarantee the accuracy and security of data and information?
* E- Efficiency. Does the system make maximum use of available resources including people, time, flow of forms, minimum processing details and the like?
* S- Services. Does the system provide desirable and reliable service to those who need it? Is the system flexible and expandable?

**HOW DO THE END-USERS/CUSTOMERS FEEL ABOUT THE SOLUTION?**

It’s important not only to evaluate whether a system can work but we must also evaluate whether a system will work. A workable solution might fail because of end-user and management resistance.

***Feasibility Study, Cost Benefit Analysis for your new eShopping Web Site***

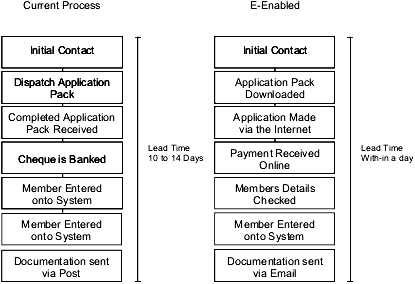
**OVERVIEW**

This case study will try and asses the feasibility for an ecommerce website.

**CHALLENGE**

The challenge in this case study is do the following

Analyze the costs and benefits associated with the implementation of an ecommerce website for a membership society.



**Fig. Membership Process**

Figure 1 illustrates typifies the processes of a generic membership system. You can see the effect e-enablement has on the current process on the right site of the illustration.

`

**Costs for technology**

To produce an e-shopping website requires a high speed connection to the Internet, a web server, and software. Other costs that are relevant is the cost of the payment system, whether it is taking online payment directly from the Societies web site or an alternative third-party like Pay pal or more expensively using an online bank.

**Costs for technological development**

This will involve a number of programmers who are able to interpret your functional requirements and program/create your website.

**Costs for the consultancy support (design and implementation)**

You would require the services of specialists in e-business design and implementation to guide you through this process.

**Costs for the organization for piloting training**

During the technological development of a website it is always a good idea to allow admin staff who will be using the system on a daily basis to pilot the system to as a training initiative.

**Running costs**

These are an upkeep of the web server and maintenance costs.

**Running costs for change process**

This is the cost of factoring in for your employees to train and adapting to the newly introduced technology, mainly the strategies used to make the change as smooth as possible.

Additionally being on the Internet would result in the your company having to become familiar to respond to emails, queries, and complaints that require instant or quick responses as opposed to replying to a Customer/Client via a letter. To be successful online, your company would have to address this issue of Change Management in that it would have to incorporate into its business, processes in order to guide the company to successfully maximise its effectiveness on the Internet. “Business is streamlined and service is almost instantaneous when it is done on the Web.

***FINANCIAL BENEFITS***

**Improve Cash flow**

Online payment would result in the lead-time to receive payments for membership with in the same day of the application being made rather than the average 14 day delay. E-enabling the membership process is not just about reengineering a process so that it is quicker than before, it would result in a complete overhaul of the previous way of managing membership. Figure 1 illustrates the current membership process on the left and the E-enabled equivalent. As you can see in the E-enabled application and payment are made online. Once the application has been completed the documentation is sent via email to the member adding value by cutting the waiting time. This also saves on printing and postage for the Society. Even if the member does not have an email address the documentation will be available for download for registered members to the web site.

**Increase Revenues**

The Internet will increase the volume of members. By going online with your business, you will generate revenue from places you never imagined”

*NON-FINANCIAL BENEFITS*

**Communication**

Direct email marketing incurs little or no cost compared to the traditional direct mail marketing. Direct email marketing allows the flexibility of sending the your companies message day or night, exactly when they want.

**Transparency**

The Information Management Website will allow the membership process to become transparent. For example, for the first time ever, Management will be able to know as a matter of fact:

* The total number of members
* Those members who need to renew their membership
* Those members who are in arrears with their membership fees
* Total number of members and accredited members
* Forecast for the expected revenue that will be generated in advance and look at historic monthly generated revenues.

**Exposure**

The Internet means that your company will become a global Business. Thus attracting potential members from internationally. “Using the Web to sell your products removes the physical boundaries from your customer base. Customers from all over the world can learn about and purchase your products online”

***Online Shopping System Process WorkFlow***

**Continue Shopping**

Clear Cart

Proceed To Checkout

**Confirmation Page**

Purchase Confirmation Details

Submit Purchase

Enter Personal Details

Purchase Calculations

**CheckOut Page**

Cart Items

Item 1

Item 2

Item 3

Item 4

**Cart Page**

**Category**

Category 1

Category 2

Category 3

Category 4

**Index Page**

View Cart

Add To Cart

**Product**

Product 1

Product 2

Product 3

Product 4

**Category Page**

***Data Flow Diagram***

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

**Symbols and Notations Used in DFDs**

Two common systems of symbols are named after their creators:

* Yourdon and Coad
* Yourdon and DeMarco
* Gane and Sarson

One main difference in their symbols is that Yourdon-Coad and Yourdon-DeMarco use circles for processes, while Gane and Sarson use rectangles with rounded corners, sometimes called lozenges. There are other symbol variations in use as well, so the important thing to keep in mind is to be clear and consistent in the shapes and notations you use to communicate and collaborate with others.

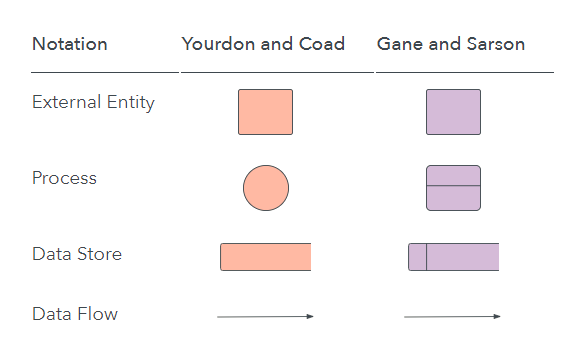
Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

**External entity:** An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

**Process:** Any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”

**Data store:** Files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”

**Data flow:** The route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

****

**DFD rules and tips**

* **Each process should have at least one input and an output.**
* **Each data store should have at least one data flow in and one data flow out.**
* **Data stored in a system must go through a process.**
* **All processes in a DFD go to another process or a data store.**

**DFD levels and layers: From context diagrams to pseudocode**

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

* DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.
* DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.
* DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system’s functioning.
* Progression to Levels 3, 4 and beyond is possible, but going beyond Level 3 is uncommon. Doing so can create complexity that makes it difficult to communicate, compare or model effectively.

Using DFD layers, the cascading levels can be nested directly in the diagram, providing a cleaner look with easy access to the deeper dive.

By becoming sufficiently detailed in the DFD, developers and designers can use it to write pseudocode, which is a combination of English and the coding language.  Pseudocode facilitates the development of the actual code.

**Examples of how DFDs can be used**

Data flow diagrams are well suited for analysis or modeling of various types of systems in different fields.

**DFD in software engineering:** This is where data flow diagrams got their main start in the 1970s. DFDs can provide a focused approached to technical development, in which more research is done up front to get to coding.

**DFD in business analysis:** Business analysts use DFDs to analyze existing systems and find inefficiencies. Diagramming the process can uncover steps that might otherwise be missed or not fully understood.

**DFD in business process re-engineering:**  DFDs can be used to model a better, more efficient flow of data through a business process. BPR was pioneered in the 1990s to help organizations cut operational costs, improve customer service and better compete in the market.

**DFD in agile development:** DFDs can be used to visualize and understand business and technical requirements and plan the next steps. They can be a simple yet powerful tool for communication and collaboration to focus rapid development.

**DFD in system structures:** Any system or process can be analyzed in progressive detail to improve it, on both a technical and non-technical basis.

**DFD vs. Unified Modeling Language (UML)**

While a DFD illustrates how data flows through a system, UML is a modeling language used in Object Oriented Software Design to provide a more detailed view. A DFD may still provide a good starting point, but when actually developing the system, developers may turn to UML diagrams such as class diagrams and structure diagrams to achieve the required specificity.

**Logical DFD vs. Physical DFD**

These are the two categories of a data flow diagram. A Logical DFD visualizes the data flow that is essential for a business to operate. It focuses on the business and the information needed, not on how the system works or is proposed to work. However, a Physical DFD shows how the system is actually implemented now, or how it will be. For example, in a Logical DFD, the processes would be business activities, while in a Physical DFD, the processes would be programs and manual procedures.

**DFD Level 0**

Vendor Management

Place Order

User Details Management

Database

Cart Management

View Product

Product Management

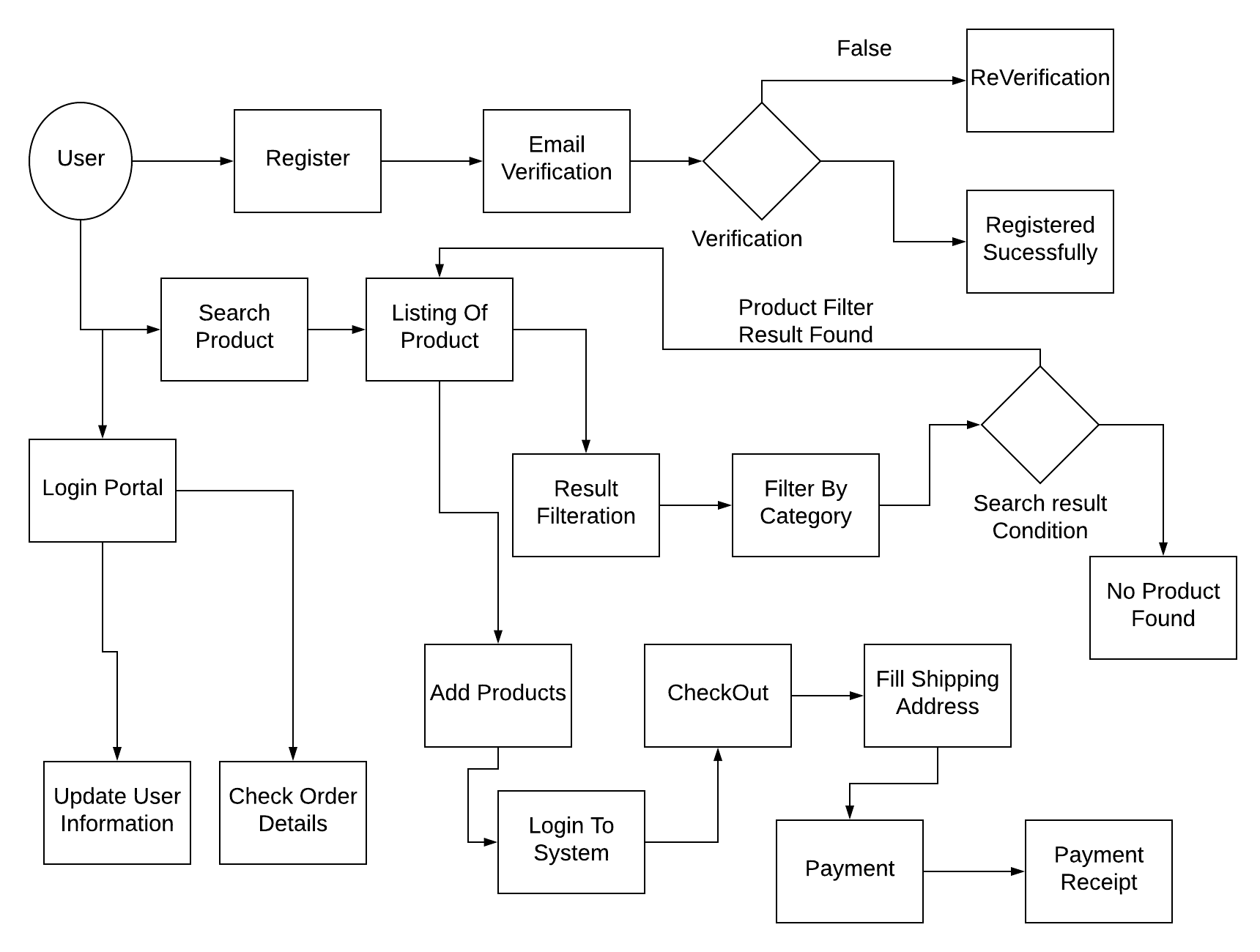
User Management

**E-commerce Portal Management**

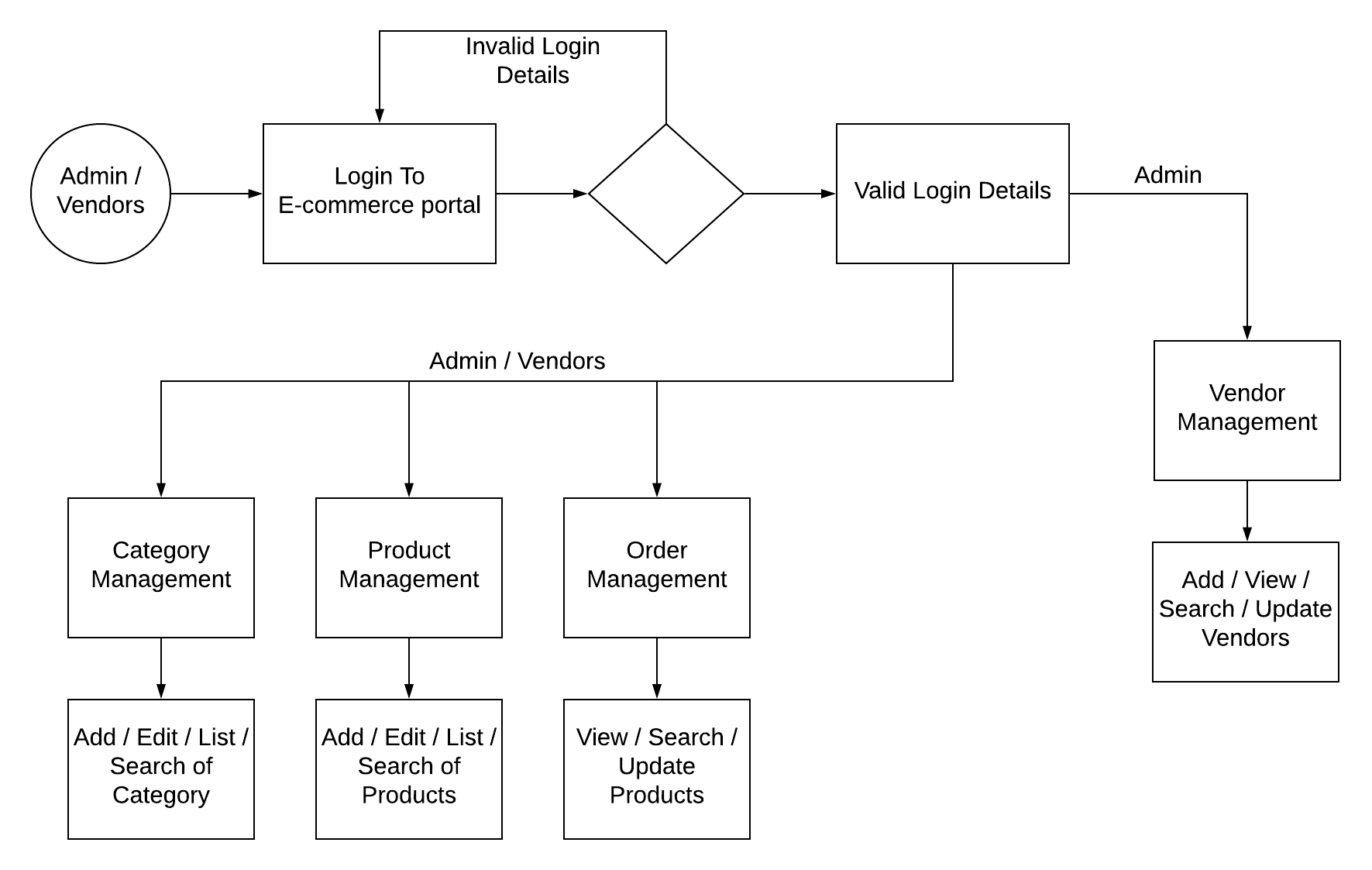
Admin/ Vendor

Users/ Customers

**DFD Level 1**

****

**DFD Level 2**

****

***Entity Relationship Diagram***

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

**Uses of entity relationship diagrams**

**Database design:** ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project. It’s also later used to model a particular database or databases. A relational database has an equivalent relational table and can potentially be expressed that way as needed.

**Database troubleshooting:** ER diagrams are used to analyze existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it’s going wrong.

**Business information systems:** The diagrams are used to design or analyze relational databases used in business processes. Any business process that uses fielded data involving entities, actions and interplay can potentially benefit from a relational database. It can streamline processes, uncover information more easily and improve results.

**Business process re-engineering (BPR):** ER diagrams help in analyzing databases used in business process re-engineering and in modeling a new database setup.

**Education:** Databases are today’s method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures.

**Research:** Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyze the data.

**The components and features of an ER diagram**

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here’s a glossary:

**Entity**

A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.

ENTITY

**Entity type:** A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars or products.

**Entity set:** Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day. Other examples: Customers who purchased last month, cars currently registered in Florida. A related term is instance, in which the specific person or car would be an instance of the entity set.

**Entity categories:** Entities are categorized as strong, weak or associative. A strong entity can be defined solely by its own attributes, while a weak entity cannot. An associative entity associates entities (or elements) within an entity set.

ASSOCIATIVE ENTITY

WEAK ENTITY

**Entity keys:** Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate or primary.

* **Super key:** A set of attributes (one or more) that together define an entity in an entity set.
* **Candidate key:** A minimal super key, meaning it has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key.
* **Primary key:** A candidate key chosen by the database designer to uniquely identify the entity set. Foreign key: Identifies the relationship between entities.

**Relationship**

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

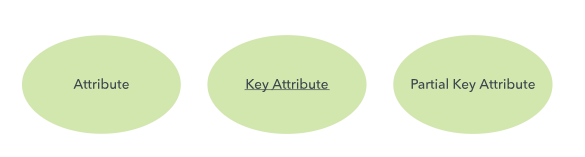
RELATIONSHIP

WEAK RELATION-SHIP

**Recursive relationship:** The same entity participates more than once in the relationship.

**Attribute**

A property or characteristic of an entity. Often shown as an oval or circle.



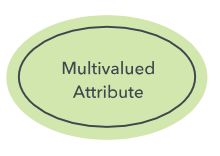
**Descriptive attribute:** A property or characteristic of a relationship (versus of an entity.)

**Attribute categories:** Attributes are categorized as simple, composite, derived, as well as single-value or multi-value.

* **Simple:** Means the attribute value is atomic and can’t be further divided, such as a phone number.
* **Composite:** Sub-attributes spring from an attribute.
* **Derived:** Attributed is calculated or otherwise derived from another attribute, such as age from a birthdate.

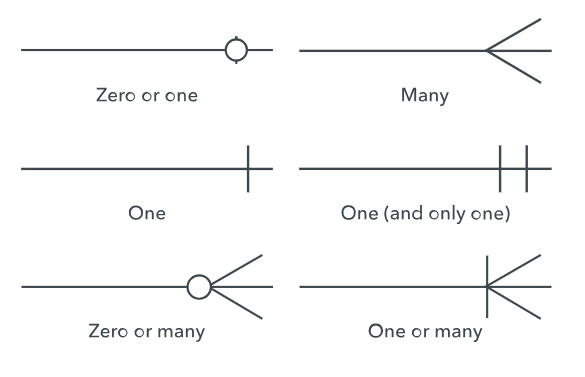


**Multi-value:** More than one attribute value is denoted, such as multiple phone numbers for a person.



**Single-value:** Just one attribute value. The types can be combined, such as: simple single-value attributes or composite multi-value attributes.

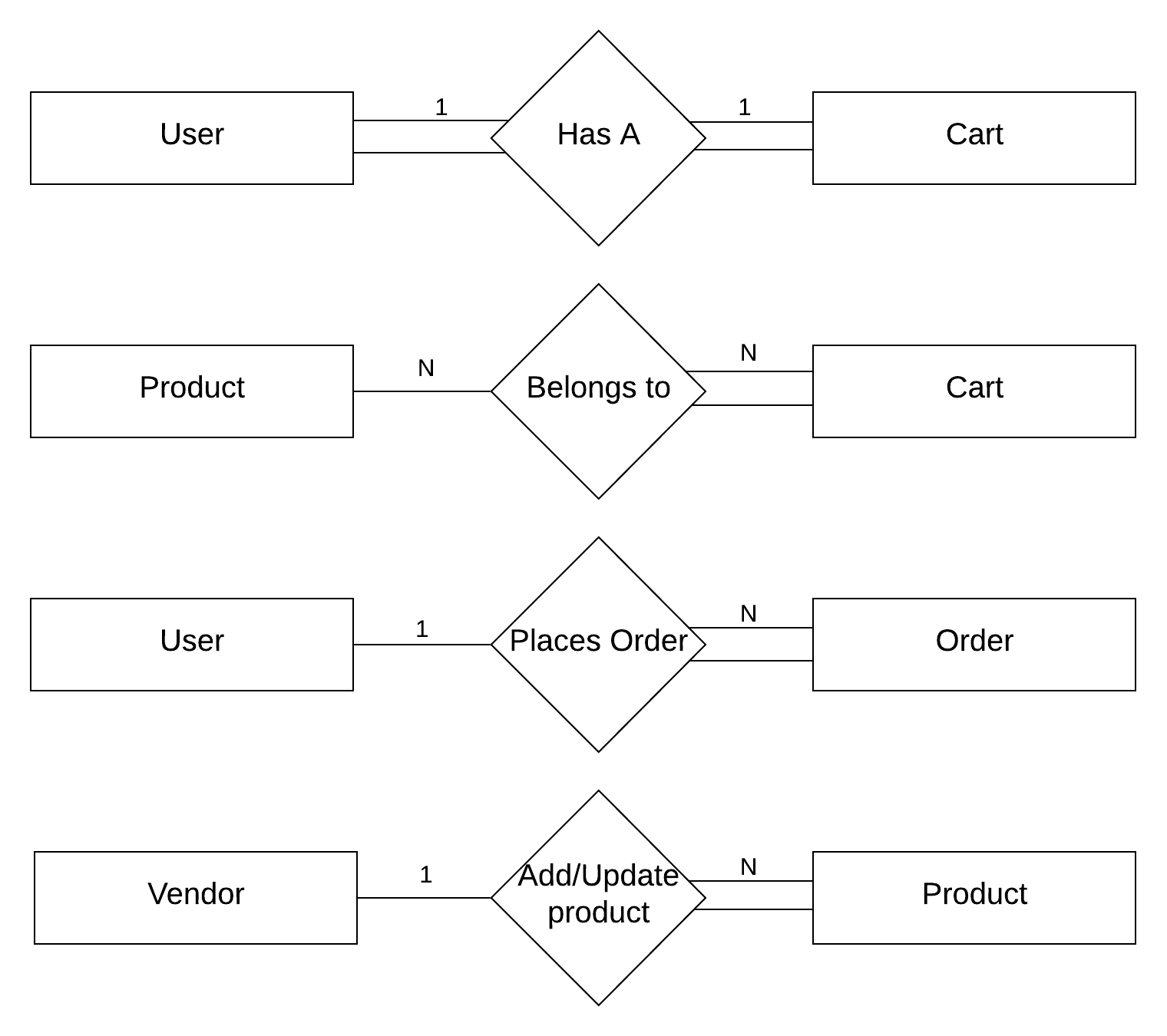
**Cardinality**

Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many. A one-to-one example would be one student associated with one mailing address. A one-to-many example (or many-to-one, depending on the relationship direction): One student registers for multiple courses, but all those courses have a single line back to that one student. Many-to-many example: Students as a group are associated with multiple faculty members, and faculty members in turn are associated with multiple students 

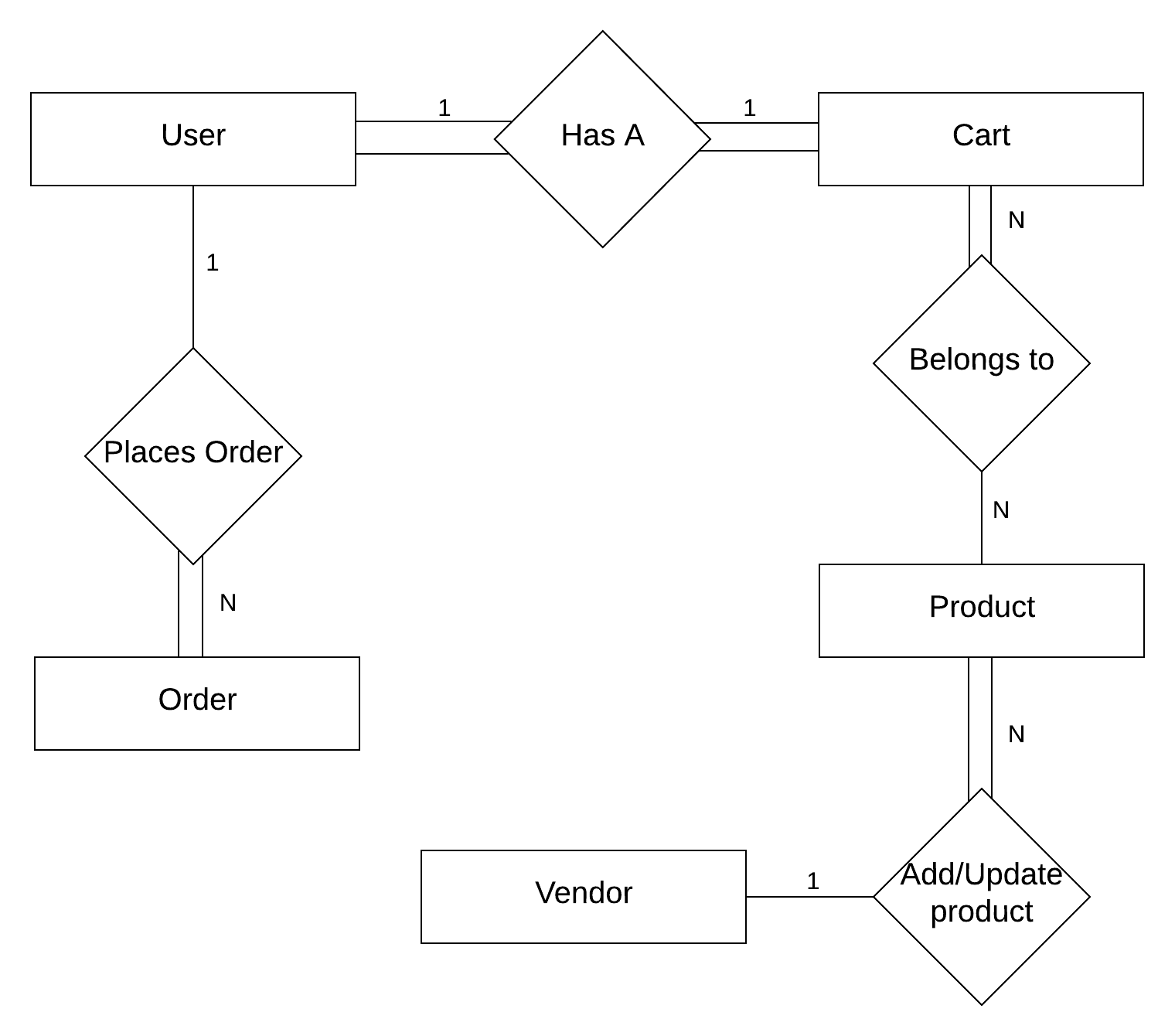
**Cardinality views:** Cardinality can be shown as look-across or same-side, depending on where the symbols are shown.

**Cardinality constraints:** The minimum or maximum numbers that apply to a relationship.

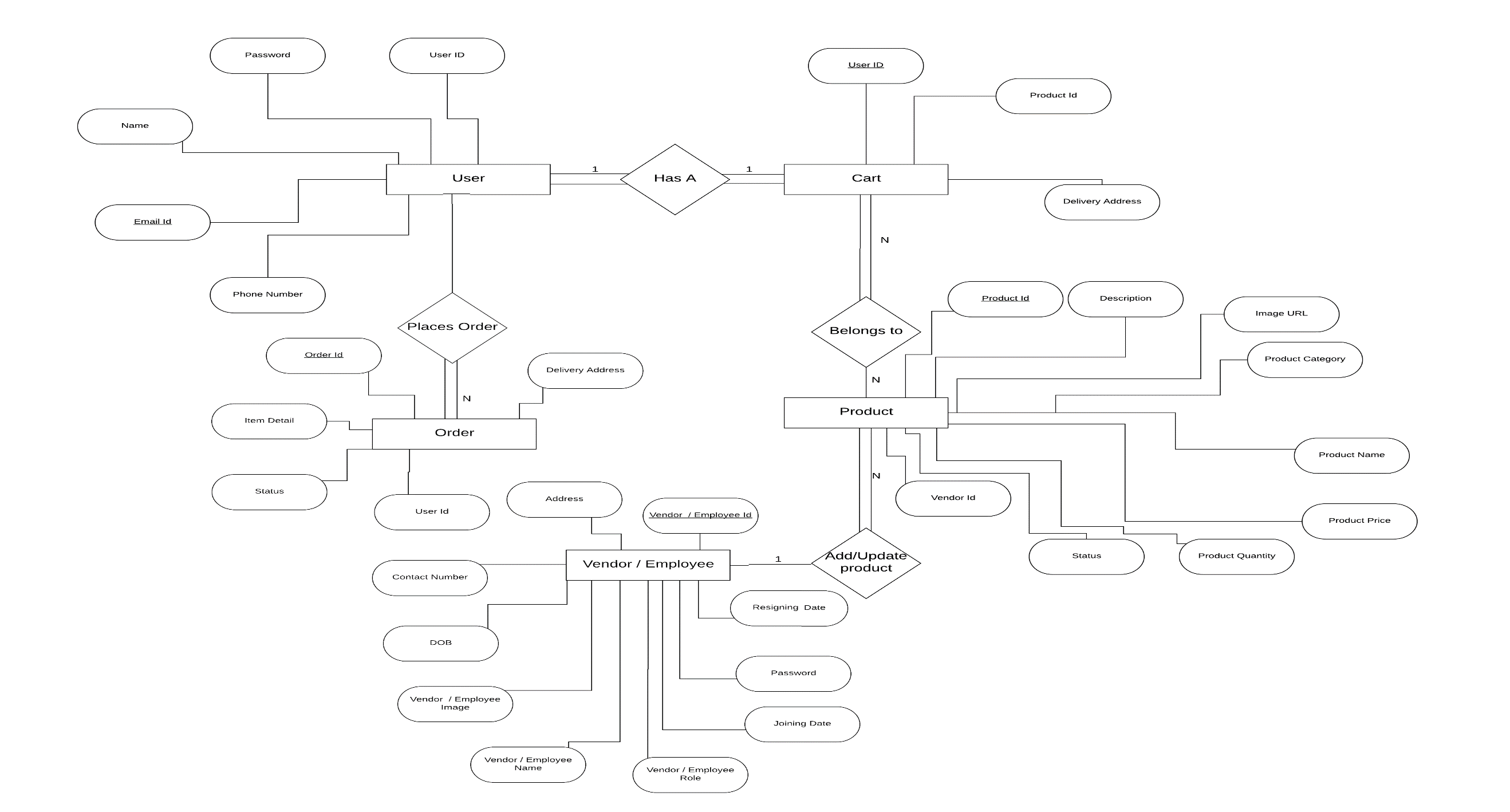
**Relationship Between Entities**



**Complete Structure of Relationship Between Entities**



**ER Diagram**



# ***Software Engineering and Software Paradigms***

The term "software engineering" was coined in about 1969 to mean "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines".

This view opposed uniqueness and "magic" of programming in an effort to move the development of software from "magic" (which only a select few can do) to "art" (which the talented can do) to "science" (which supposedly anyone can do!). There have been numerous definitions given for software engineering (including that above and below).

Software Engineering is not a discipline; it is an aspiration, as yet unachieved. Many approaches have been proposed including reusable components, formal methods, structured methods and architectural studies. These approaches chiefly emphasize the engineering product; the solution rather than the problem it solves.

Software Development current situation:

* People developing systems were consistently wrong in their estimates of time, effort, and costs
* Reliability and maintainability were difficult to achieve
* Delivered systems frequently did not work
* 1979 study of a small number of government projects showed that:
* 2% worked
* 3% could work after some corrections
* 45% delivered but never successfully used
* 20% used but extensively reworked or abandoned
* 30% paid and undelivered
* Fixing bugs in delivered software produced more bugs
* Increase in size of software systems
* NASA
* StarWars Defense Initiative
* Social Security Administration
* financial transaction systems
* Changes in the ratio of hardware to software costs
* early 60's - 80% hardware costs
* middle 60's - 40-50% software costs
* today - less than 20% hardware costs
* Increasingly important role of maintenance
* Fixing errors, modification, adding options
* Cost is often twice that of developing the software
* Advances in hardware (lower costs)
* Advances in software techniques (e.g., users interaction)
* Increased demands for software
* Medicine, Manufacturing, Entertainment, Publishing
* Demand for larger and more complex software systems
* Airplanes (crashes), NASA (aborted space shuttle launches),
* "ghost" trains, runaway missiles,
* ATM machines (have you had your card "swallowed"?), life-support systems, car systems, etc.
* US National security and day-to-day operations are highly dependent on computerized systems.

Manufacturing software can be characterized by a series of steps ranging from concept exploration to final retirement; this series of steps is generally referred to as a *software lifecycle*.

Steps or phases in a software lifecycle fall generally into these categories:

* Requirements (Relative Cost 2%)
* Specification (analysis) (Relative Cost 5%)
* Design (Relative Cost 6%)
* Implementation (Relative Cost 5%)
* Testing (Relative Cost 7%)
* Integration (Relative Cost 8%)
* Maintenance (Relative Cost 67%)
* Retirement

Software engineering employs a variety of methods, tools, and paradigms.

Paradigms refer to particular approaches or philosophies for designing, building and maintaining software. Different paradigms each have their own advantages and disadvantages which make one more appropriate in a given situation than perhaps another (!).

A method (also referred to as a technique) is heavily depended on a selected paradigm and may be seen as a procedure for producing some result. Methods generally involve some formal notation and process(es).

Tools are automated systems implementing a particular method.

Thus, the following phases are heavily affected by selected software paradigms

* Design
* Implementation
* Integration
* Maintenance

The software development cycle involves the activities in the production of a software system. Generally the software development cycle can be divided into the following phases:

* Requirements analysis and specification
* Design
* Preliminary design
* Detailed design
* Implementation
* Component Implementation
* Component Integration
* System Documenting
* Testing
* Unit testing
* Integration testing
* System testing
* Installation and Acceptance Testing
* Maintenance
* Bug Reporting and Fixing
* Change requirements and software upgrading

Software lifecycles that will be briefly reviewed include:

1. Build and Fix model
2. Waterfall and Modified Waterfall models
3. Rapid Prototyping
4. Boehm's spiral model

## Build and Fix model

This works OK for small, simple systems, but is completely unsatisfactory for software systems of any size. It has been shown empirically that the cost of changing a software product is relatively small if the change is made at the requirements or design phases but grows large at later phases.

The cost of this process model is actually far greater than the cost of a properly specified and designed project. Maintenance can also be problematic in a software system developed under this scenario.

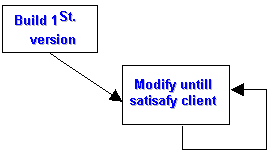


Figure: Build and Fix model

## Waterfall and Modified Waterfall models

### Waterfall Model

Derived from other engineering processes in 1970. Offered a means of making the development process more structured. Expresses the interaction between subsequent phases.

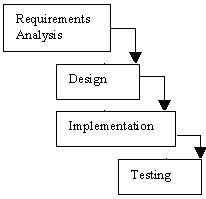


Figure: Waterfall model

Each phase cascades into the next phase. In the original waterfall model, a strict sequentially was at least implied. This meant that one phase had to be completed before the next phase was begun.

It also did not provide for feedback between phases or for updating/re-definition of earlier phases. Implies that there are definite breaks between phases, i.e., that each phase has a strict, non-overlapping start and finish and is carried out sequentially.

Critical point is that no phase is complete until the documentation and/or other products associated with that phase are completed.

### Modified Waterfall Model

Needed to provide for overlap and feedback between phases. Rather than being a simple linear model, it needed to be an iterative model. To facilitate the completion of the goals, milestones, and tasks, it is normal to freeze parts of the development after a certain point in the iteration. Verification and validation are added. Verification checks that the system is correct (building the system right). Validation checks that the system meets the users desires (building the right system).

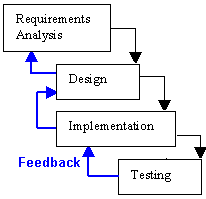


Figure: Modified Waterfall model

The waterfall model (and modified waterfall model) are inflexible in the partitioning of the project into distinct phases. However, they generally reflect engineering practice.

Considerable emphasis must be placed on discerning users' needs and requirements prior to the system being built. The identification of users' requirements as early as possible, and the agreement between user and developer with respect to those requirements, often is the deciding factor in the success or failure of a software project. These requirements are documented in the requirements specification, which is used to verify whether subsequent phases are complying with the requirements. Unfortunately specifying users' requirements is very much an art, and as such is extremely difficult. Validation feedback can be used to prevent the appearance of a strong divergence between the system under development and the users' expectations for the delivered system. Unfortunately, the waterfall lifecycle (and the modified waterfall lifecycle) are inadequate for realistic validation activities. They are exclusively document driven models. The resulting design reality is that only 50% of the design effort occurs during the actual design phase with 1/3 of the design effort occurring during the coding activity! This is topped by the fact that over 16% of the design effort occurs after the system is supposed to be completed! In general the behavior of many individuals in this type of process is opportunistic. The boundaries of phases are indiscriminately crossed with deadlines being somewhat arbitrary.

## Rapid Prototyping

Prototyping also referred to as evolutionary development, prototyping aims to enhance the accuracy of the designer's perception of the user's requirements. Prototyping is based on the idea of developing an initial implementation for user feedback, and then refining this prototype through many versions until an satisfactory system emerges. The specification, development and validation activities are carried out concurrently with rapid feedback across the activities. Generally, prototyping is characterized by the use of very high-level languages, which probably will not be used in the final software implementation but which allow rapid development, and the development of a system with less functionality with respect to quality attributes such as robustness, speed, etc.

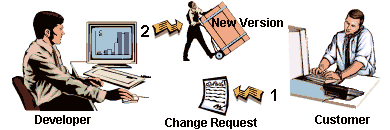


Figure: Rapid Prototyping model

Prototyping allows the clarification of users requirements through, particularly, the early development of the user interface. The user can then try out the system, albeit a (sub) system of what will be the final product. This allows the user to provide feedback before a large investment has been made in the development of the wrong system.

There are two types of prototypes:

* **Exploratory programming:** Objective is to work with the user to explore their requirements and deliver a final system. Starts with the parts of the system which are understood, and then evolves as the user proposes new features.
* **Throw-away prototyping:** Objective is to understand the users' requirements and develop a better requirements definition for the system. Concentrates on poorly understood components.

Experiments with prototyping showed that this approach took 40% less time and resulted in 45% less code; however, it produced code which was not as robust, and therefore more difficult to maintain. Documentation was often sacrificed or done incompletely. The schedule expectations of users and managers tended to be unrealistic especially with respect to throw-away prototypes.

## Boehm’s Spiral Model

Need an improved software lifecycle model which can subsume all the generic models discussed so far. Must also satisfy the requirements of management.

Boehm proposed a spiral model where each round of the spiral

1. a) identifies the sub problem which has the highest risk associated with it
2. b) finds a solution for that problem.

***SYSTEM DESIGN***

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design specifications to performance specification. System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs (sources), out puts (destinations), databases (data sores) and procedures (data flows) all in a format that meats the uses requirements. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

**LOGICAL DESIGN :**

Logical design of an information system shows the major features and also how they are related to one another. The first step of the system design is to design logical design elements. This is the most creative and challenging phase and important too. Design of proposed system produces the details of the state how the system will meet the requirements identified during the system analysis that is, in the design phase we have to find how to solve the difficulties faced by the existing system. The logical design of the proposed system should include the details that contain how the solutions can be implemented. It also specifies how the database is to be built for storing and retrieving data, what kind of reports are to be created and what are the inputs to be given to the system. The logical design includes input design, output design, and database design and physical design.

**INPUT DESIGN:**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing data entry. The activity of putting data into the computer for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling errors, avoiding delay, avoiding extra steps and keeping the process simple.

The system needs the data regarding the asset items, depreciation rates, asset transfer and physical verification for various validation, checking, calculation and record generation. The error raising method is also included in the software, which helps to raise error message while wrong entry of input is done. So in input design the following things are considered.

What data should be given as input?

* How the data should be arranged or coded?
* The dialogue to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.
* The samples of screen layout are given in the appendix.

**OUTPUT DESIGN :**

Computer output is the most important and direct information source to the user. Output design is a process that involves designing necessary outputs in the form of reports that should be given to the users according to the requirements. Efficient, intelligible output design should improve the system's relationship with the user and help in decision making. Since the records are directing referred by the management for taking decisions and to draw conclusions they must be designed with almost care and the details in the records must be simple, descriptive and clear to the user. So while designing output the following things are to be considered.

* Determine what information to present.
* Arrange the presentation of information in an acceptable format.
* Decide how to distribute the output to intended receipts.

Depending on the nature and future use of output required, they can be displayed on the monitor for immediate need. All screens are informative and interactive in such a way that the user can full fill his requirements through asking queries.

**PHYSICAL DESIGN**

The process of developing the program software is referred to as physical design. We have to design the process by identifying reports and the other outputs the system will produce. Coding the program for each module with its logic is performed in this step. Proper software specification is also done in this step.

**MODULAR DESIGN**

A software system is always divided into several sub systems that makes it easier for the development. A software system that is structured into several subsystems makes it easy for the development and testing. The different subsystems are known as the modules and the process of dividing an entire system into subsystems is known as modularization or decomposition.

A system cannot be decomposed into several subsystems in any way. There must some logical barrier, which facilitates the separation of each module. The separation must be simple but yet must be effective so that the development is not affected.

The system under consideration has been divided into several modules taking in consideration the above-mentioned criteria.

***HUMAN RESOURCES***

The most important factor for any organization to keep abreast their project is to build a good employee management relationship. Working on the internal and external environment needs heavy amount of work to be performed in this sector. The resources for hiring good employees is categories as follows Local Newspaper and Media will be use to advertise. Such media aswww.rozee.pk and [www.brightspree.com](http://www.brightspree.com) for keeping track of the project Such Designation will be required

1) Network Administrator

2) Database Administrator

3) Web base Designers and Developers

4) Production Departmentt

5) Strategist

***IMPACT ON THE BUSINESS***

Impact of business is entirely depend on its internal and external factors. It is the aim of the organization to cover all their weaknesses with strength and take available opportunities and remove threats. Once the project completed it the entirely the Strategist to work on the factor will help improvising necessary terminologies /technologies in order to be effective. The following points illustrate the data we used to collect information in the system analysis phase.

* Learn all we can from existing documents, forms, reports and databases.
* If appropriate we will observe the system in action. We agree not to ask questions but just take notes and draw pictures. We must make sure that the workers know we are not evaluating individuals. Otherwise they may tend to work more efficiently than they usually do.
* Given all the facts we have already collected. We will design and distribute questionnaires to clear things up we don’t fully understand. This is also a good time to solicit opinions on problems and limitations. We understand that questionnaires do take much of the end-user time but they must know when its best to make that sacrifice.
* Conduct our interviews to verify and clarify the most important and difficult issues and problems.
* Follow up. We will use appropriate fact-finding techniques to verify facts, usually through interviews and observation.

**SYSTEM IMPLEMENTATION**

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively.

There are several activities involved while implementing a new project. They are

* End user training
* End user Education
* Training on the application software
* System Design
* Parallel Run and To New System
* Post implementation Review

**End user Training:**

The successful implementation of the new system will purely upon the involvement of the officers working in that department. The officers will be imparted the necessary training on the new technology

**End User Education:**

The education of the end user start after the implementation and testing is over. When the system is found to be more difficult to understand and complex, more effort is put to educate the end used to make them aware of the system, giving them lectures about the new system and providing them necessary documents and materials about how the system can do this.

**Training of application software:**

After providing the necessary basic training on the computer awareness, the users will have to be trained upon the new system such as the screen flows and screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the way to correct the data entered. It should then cover information needed by the specific user or group to use the system.

**Post Implementation View:**

The department is planning a method to know the states of the past implementation process. For that regular meeting will be arranged by the concerned officers about the implementation problem and success.

***Software Testing***

Software testing is a process of executing a program or application with the intent of finding the software bugs.

It can also be stated as the process of validating and verifying that a software program or application or product:

* Meets the business and technical requirements that guided it’s design and development
* Works as expected
* Can be implemented with the same characteristic.

Let’s break the definition of Software testing into the following parts:

1. Process: Testing is a process rather than a single activity.
2. All Life Cycle Activities: Testing is a process that’s take place throughout the Software Development Life Cycle (SDLC).

* The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it’s referred as “verifying the test basis via the test design”.
* The test basis includes documents such as the requirements and design specifications.
* **Static Testing:** It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.
* **Dynamic Testing:**  In dynamic testing the software code is executed to demonstrate the result of running tests. It’s done during validation process. For example: unit testing, integration testing, system testing, etc.
* **Planning:** We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.
* **Preparation:**  We need to choose what testing we will do, by selecting test conditions and designing test cases.
* **Evaluation:** During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.
* **Software products and related work products:** Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.