***SDLC***

Name: Abdelkareem Yousef Soubar

Table of Contents

XXXX

# **Introduction**

In this short section I would like to talk about what is SDLC and what are the benefits of it. SDLC stands for System Development Life Cycle, its main task is to provide software by following certain steps that enable the software to function at its optimum state.

## **Steps for SDLC**

You may ask yourself what are these main steps that are involved in SDLC, I would like to answer your question in this paragraph.

1. **Project Planning:** In this phase, we answer the number one question in creating software, which is “What do we want?”. Once this question is answered, we can start with further phases. It is essential that we know what we want, because if there is some uncertainty within the idea of the project, I can assure you that it will be hard to proceed with it further.
2. **Gathering Requirements and Analysis:** We meet with the customer to find out what are the functional requirements and what are the nonfunctional requirements. In brief, functional requirements answer the question “How must the system work?” and nonfunctional requirements would answer the question “How should the system perform?”.
3. **Design:** In the design phase we transform data that was gathered from the previous two phases into a prototype if you can call it that way. A design can include wireframes, UML diagrams, flowcharts, etc.…
4. **Coding/Implementation:** After getting set with all of the information that we have, coding takes place. Coding is considered to be one of the longest phases in SDLC, this is because we must ensure the integrity of our software and follow certain design guidelines that are within the professional industry standards as well as the company’s standards.
5. **Testing:** No software is complete without having it tested. Tests can be automated or manual, in which a whole quality assurance team would be hired to test the software. This phase is very important, in which it can show some flaws that were not thought of as a programmer’s point-of-view.
6. **Deployment:** This phase is all about deploying the produced software to the customer. It is really necessary to get feedback from the customer and follow up with him/her.

Of course, the software that is deployed successfully met all of the requirements with the tests passed.

1. **Maintenance:** If any bug has been discovered by the customer, he/she will inform us as soon as possible. A certain bug could show from a customer’s point-of-view not a programmer’s point-of-view, since a programmer didn’t really think that much regarding this bug or how it could be produced.

## **Benefits of SDLC**

There are many benefits of using SDLC, I would like to point out a few!

* It makes all parties engaged in the development process more aware of every stage of the life cycle.
* It facilitates project management and control.
* It helps with project estimating, planning, and scheduling.
* Project hazards are decreased.

# **Lifecyle Models**

In this part of this report, I would like to discuss two main categories of software lifecycle models which are iterative and sequential models. I will talk about the properties of each one and also give examples of them.

## **Iterative Lifecyle Model**

A procedure where development is carried out in brief, repeating cycles, or iterations, is known as an iterative software development lifecycle. The goal is to quickly provide a functional version of the program so that it may be enhanced in later iterations. Planning, design, development, testing, and deployment processes are all included in each iteration. Up till the desired program is created, the procedure is repeated. Throughout the process, input is gathered, and the software is continually enhanced in response to it. With this strategy, it is possible to be flexible and alter courses as necessary.

There are two main examples that are under the tree of iterative lifecycle model, agile and scrum.

### **Agile**

Agile is an iterative development process that places an emphasis on adaptability, teamwork, and quick iteration. Agile teams complete a tiny piece of functionality's planning, design, development, testing, and deployment in brief sprints, which are generally 2-4 weeks long. The team then evaluates the sprint's work, collects feedback, and utilizes it to plan the following sprint. Until the full piece of software is created, this procedure is repeated.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Customer satisfaction is rapid. | It is not a document-based method for creating software. |
| The product is frequently delivered. | Since this model can adapt to changes, it is very unclear to predict what the outcome would be. |
| Regular tolerance to shifting conditions. | Estimating the amount of work and resources is challenging. |



### **Scrum**

Scrum is an iterative method that focuses on managing iterative development rather than specific agile practices. Because it is easy to use, adaptable, and produces results rapidly, it is especially well-liked in the software development industry. It uses diverse teams to quickly iterate through functioning software to build (sprints).

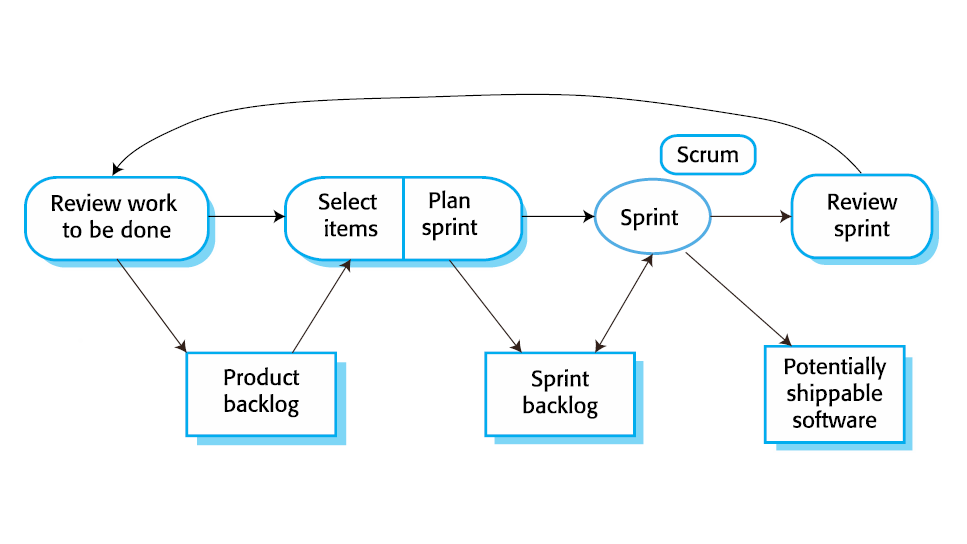
There are three main phases when using the scrum method in SDLC, I would like to give an outline to them.

* The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
* After this step, it is followed by a series of sprint cycles in which each sprint could take two to four weeks to accomplish. Each cycle develops an “increment” of the system.
* The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and also assesses the lessons learned from the project.

In the following table, I would like to discuss some key regarding the scrum method in software lifecycle.

|  |  |
| --- | --- |
| **Scrum Term** | **Definition** |
| **Scrum** | It is a daily meeting that reviews the progress and prioritizes work to be done that day |
| **Scrum Master** | The Scrum Master is responsible for ensuring that the scrum process is followed and guides the team in the effective use of scrum |
| **Product Backlog** | It is a list of ‘to do’ items that the scrum team must handle and solve. |
| **Product Owner** | An induvial or a small group whose job is to identify product features or requirements. |

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| The scrum methodology ensures the integrity of using time and money resources. | To be in the scrum framework, intensive experience and training must be involved, making it sometimes hard and expensive. |
| It is a good option for projects with interchanging variables and conditions. | When implementing this style, project deadlines must not be strict. |
| Due to having scrum meetings, progress is well tracked. | Scrum meetings can sometimes affect people working on the project in a negative aspect. |



## **Sequential Lifecycle Model**

A sequential lifecycle is the traditional way of software development, in which they are divided into certain phases. You can classify a sequential lifecycle as a linear model, this is because you cannot go back to a previous phase. If you want to go to the next phase, you need to complete the phase that is prerequisite to it. This type of model is very useful when the project requirements are well-defined and there are no variables to be changed. However, if there are any changing variables, it is very hard to change them, since this model cannot tolerate any flexibility.

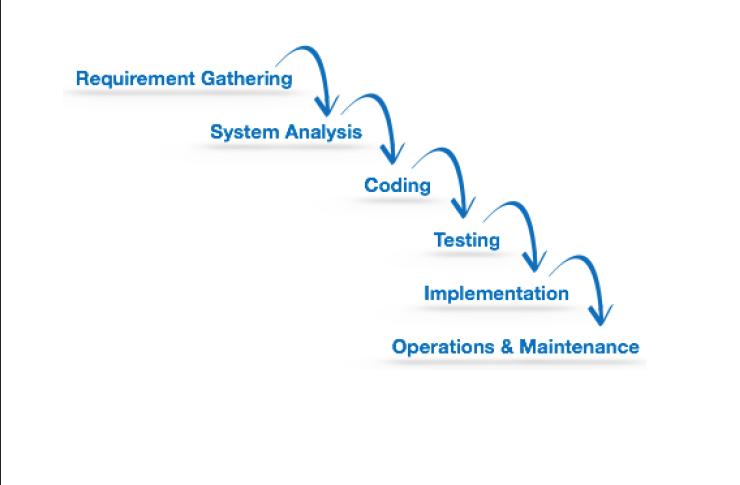
I would like to address two main examples that are under the sequential lifecycle categories. (Waterfall and V Models)

### **Waterfall Model**

In the early stages of the software engineering era, the waterfall was the most commonly used method, but nowadays it is not that commonly used. However, this model is the basis of every other model that exists. This model is quite easy and convenient to use and understand. There are several reasons that make the waterfall model easy to use and understand.

* **Linear and Sequential:** Since the waterfall model is linear and sequential, it would provide a comprehensible and logical progression of steps. This is because each phase of the development process must be completed before moving on to the next phase.
* **Document-Driven:** Unlike scrum, this model relies heavily on documentation. Documentation really helps with making things clear and keeping track of every step that is produced.
* **Suitable for Well-defined Projects:** In an environment where each phase is clear and well-defined, where variables are not prone to any variations, it is more preferred to use the waterfall model.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Strict planning of the project development structure ensures that the number of obstacles that can be faced is decreased | If any of the project requirements need to be changed, it is very hard to go back and modify it. |
| Due to heavy documentation, any information related to the project is documented. This makes it easy for any new members to get track of what is going on. | Unlike agile and scrum methodologies, no prototypes can be provided to the customer during the phases. So, the customer’s feedback is not well thought off. |
| The paradigm is strict, making management easy because each phase has distinct deliverables and a review process. | It can provide a high risk of uncertainty, this is because if a mistake is made during one of the phases, it cannot be resolved in the next phase! |



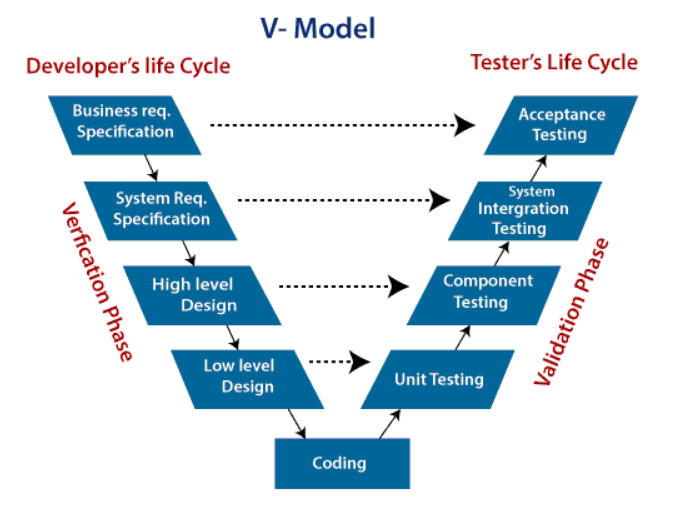
### **V Model**

V-Model is one of the famous types of sequential lifecycle models. It is considered as a strict methodology when creating certain software because of its sequential and highly structured characteristics. You may ask yourself, why is it called a V-Model? The answer to that question is that each phase of the development process corresponds to a specific stage in the product’s lifecycle. The V-model is employed to make sure that every step required to create a high-quality software product that satisfies the needs of the end-user is taken.

There are two main phases in the V-Model, in which they’re the verification and the validation phase. They are next to each other, forming the letter V. Verification involves static analysis. Static analysis is a method of debugging without actually running the program. The verification definition is that the process of evaluating the product development process is done to see whether the requirements are met. On the other hand, validation involves a dynamic analysis method, in which it means testing and evaluating is done while the software is running. Validation is the step in which it can verify that the software has met the customer’s expectations and requirements.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| The model is easy to manage since each phase is well-defined and clear. | Since changes in the requirements is hard, it makes it unsuitable to make any other changes, making it inflexible and strict. |
| It is faster than the waterfall model, because the V-Model consumes less time to develop and come out with outcomes in the development cycle. | Due to the lack of providing prototypes to the customer, it sometimes can be challenging and hard to meet the exact customer’s expectations. This can be caused due to any miscommunication by either parties. |
| Advanced error tracking is within the basis of the V-Model, which helps the people that are responsible in this project provide a bug-free product while consuming the least time possible. | If any changes are needed within the software, every document that is involved within the project must be rewritten. |

As the V-Model is a little bit more complicated than the typical waterfall model, I would like to explain its diagram in brief.



As the graphic above illustrates, we have two phases which are verification and a validation phase.

The verification phase has the following steps:

* **Business Requirement Specification:** In this step, product requirements are discussed with the customer, clear communication must be a part within meetings.
* **System Requirement Specification:** System engineers examine and study the proposed system by carefully reading the user requirement document.
* **High Level Design:** This phase can be also called “Architecture Design”, within this stage, it is clear how would the data transfer and communication between internal modules and other systems be processed.
* **Low Level Design:** This phase can be also called “Module Design”, in which the system is divided into small “modules”. Detailed internal design for all system modules is specified, it is very crucial that the design is appropriate to be used with other modules and external entities.
* **Coding Phase:** After getting done with the High Level Design and Low Level Design phases, we start with the coding phase. A certain programming language is determined with guidelines set.

After we are done with the verification phase, I would like to talk about the validation phase.Keep in mind that every step in the validation phase is parallel to a corresponding verification phase.

* **Unit Testing:** Unit testing is present within the Low Level Design phase (Module Design Phase). Tests are provided in the code level, in which it can help eliminate bugs that are present. But one thing to keep in mind is that not all bugs are eliminated in this stage.
* **Component Testing:** Component testing can be also named as “Integration Testing”, is present within the High Level Design phase (Architecture Design Phase). The main function of component testing is to test out the communication between internal modules and to verify that every is up to standard.
* **System Integration Testing:** As illustrated in the figure above, this phase corresponds directly to the System Requirement Speciation Phase. This type of testing would examine the entire system functionality as well as communication with other external systems. This phase is tested by the customer’s business team.
* **Acceptance Testing:** Finally, this phase corresponds to the Business Requirement Phase. The test is run by the point-of-view of the user’s atmosphere. Non-functional requirements-related issues can also be shown, such as the performance of the system.

# **Risk Management**

To begin with, we should define what are risks? Risks are anything that can negatively impact the flow of our project that can have a harmful effect. Risk management is a very important topic to discuss. It is not only important in SDLC, but it is also important in other businesses. In this part of the report, I would like to talk about how risks are assessed and handled.

Risk management has some tasks that are followed, with these steps and knowing each step what it does, I can assure you a successful risk management plan.

* Identify risks and their triggers.
* Classify and prioritize all risks.
* Create a plan that links each risk to mitigation.
* Monitor for risk triggers.
* Implement the mitigating action.
* Communicate risk status

Risks can be defined as the following, here is a few:

* **New Unproven Technologies**: In this era, technology has evolved in an inflated manner. And let us be honest, we all like to implement new technologies. But if there was not enough knowledge and enough documentation within a new technology, this can lead to the failure of the project.
* **Application and System Architecture:** If we chose an inadequate system architecture regarding a certain project, this could decrease the functionality of the software, making it a risk that should be avoided.
* **Organizational:** Sometimes the atmosphere when working on a certain project can tension up. This can lead to a decrease in the workflow of the project, thus taking it longer to complete it. Also, another point to consider is that the quality of the written code can also decrease, which would cost more effort, time, and money to fix it and the reputation of the endorsement could be affected also.

## **Risk Assessment**

Risk assessment is the process to identify potential hazards and analyze what could happen if a hazard occurs. We have several risk assessment techniques that can be used to identify risks in which we can decrease the severity of their effect.

* **Threat Modelling:** This entails detecting possible dangers to the software project and assessing their likelihood and effect.
* **Risk Analysis:** In this stage, we would analyze the likelihood and the impact of what a risk could negatively affect our project process.
* **Risk Response Planning:** This stage is all about how we would control and try to solve upcoming risks using the optimum techniques that would be provided.
* **Risk Monitoring and Control:** The monitoring process of any upcoming risks are included in this stage.

## **Treating Risks**

Risks can be treated in many several ways, this plays a factor on the severity of the risk’s impact, the budget of the customer, and the likelihood of risk’s existence. I would like to talk about some ways and methods that we can mitigate risks.

* **Risk Avoidance:** Sometimes if the likelihood of such a risk is apparent, we could change some properties of our project that can decrease the likelihood of that risk.
* **Risk Transfer:** If a certain organization does not have the certain capabilities to handle a certain type of risk, risk transfer can be used. This means that if a risk happens, we can get assistance from another party that could help resolve this type of risk.
* **Risk Control:** This type of risk treatment would mean taking several actions that would help prevent the escalation of that risk and its consequences.
* **Risk Acceptance:** By its name, we would accept that plan and try to develop a certain plan that would help mitigate the risk. We wouldn’t change any properties of the project, but a plan would be made to assist us in resolving the risk.

discuss using example(s) a particular lifecycle model for a specific development environment.

# **Using the Waterfall Model in Large Projects**

Although we discussed a lot of different ways and methodologies that enable us to perform software development in an efficient manner, the waterfall model stays one of the most used in large software development projects, why is that? In this chapter I would like to talk about the usage of the waterfall model in large scales.

Based on what was mentioned in a few chapters, let us have a recap on the properties of the waterfall model. As we know, the waterfall model is a sequential type of model, it means that we cannot go back to a previous step, nor we can get to the next step unless we finished the step we are in. XXXXXX

The waterfall model is used in large projects for the following reasons:

* **Clear Structure:**  By this statement, I mean that the steps that we would follow in this model are very clear. We have predefined steps that we would follow and that’s it, the steps are as followed: (Requirement Gathering, System Analysis, Coding, Testing, Implementation, Operations and Maintenance).
* **Well Documented:**  Unlike the agile methodology, the waterfall model relies heavily on documentation. This is a good method of keeping track of what is going on through our project and following what technologies and design patterns are we using. Also, if we face a problem throughout the testing process, we would definitely know how to solve it, this is because our project processes have been well-documented.
* **No Confusion:** What I would mean by the no confusion statement, is that there would be any uncertainty within the phases that we are in. For example, we would be in the coding phase, it wouldn’t be suitable that we would jump to the next phase (testing phase) while we still didn’t finish coding for the project.
* **Easily Managed:** When we have a project on a large scale, we would involve a lot of employees from different expertise to construct this project. The waterfall model enables us to follow these phases in a clear manner, making it possible to be followed by nearly everyone in the foundation.

Your report should describe how technical solution can be compared (compare two).

# **The Feasibility Report**

To start things off, we need to define what is a feasibility study. A feasibility study is done to determine whether the project that is to be started is feasible. What we mean by a project being feasible is that we can produce an outcome without having too many difficulties while producing it or is it beneficial to start working with it.

## **Feasibility Report Components**

To write a well-documented feasibility report, we have a set of essential components that should be involved in order to write a successful feasibility report.

* **Project Description:** This is a very important step to begin with, this would include the purpose of our project. It is very important to know the purpose of our project, this is because it helps us to get an idea of what our project is. Also when knowing what is the purpose of our project, it can give out a higher performance when working on our project.
* **Market Analysis:** A market analysis is performed to know the target audience and how they would react to our project. It would show how successful our project would be if implemented to the general public.
* **Technical Specifications:** What we mean by technical specifications is mentioning all of the technical details that we used. This can include the programming languages that we used, which architecture method approach was used, and different security measures.
* **Risk Analysis:** As mentioned in a previous section in this report, risk analysis is one of the most important things to do in a project. Risk analysis is used to inform the customers of certain risks that we can face throughout the process of creating certain software and how we could mitigate those apparent risks.
* **XXXX**
* **XXXX**
* [**https://www.projectpractical.com/steps-to-prepare-feasibility-study-report-for-software-development-project/**](https://www.projectpractical.com/steps-to-prepare-feasibility-study-report-for-software-development-project/)
* **Conclusion:** We must write a summary of the purpose of the project as well XXXXX

## **Feasibility Report Purpose**

There are various advantages when creating a feasibility report. Here is a few:

* It would show the advantages and disadvantages when creating a certain project. In which a feasibility report would show us details of a project in a summarized form rather than going deeply in analyzing and going through it.
* A feasibility report could show if the project is risky or not. If the project had a high-risk factor, this could help us not to start with it. If we started with this project and it failed we would have lost some precious time, effort, and money that we could have used on another more successful project.
* A good feasibility report would also be used to convince investors to invest in our product. This could mean that we would have more financial resources to help us improve our product.

## **Different Feasibility Criteria**

When conducting a feasibility study, we should include certain key aspects that would make our feasibility report a successful one.

* **Economic Feasibility:** Every project has a budget. Can this project be made with this budget or not? How good would this software be? These are some questions to ask yourself we are conducting the economic feasibility. Economic Feasibility would have the following parts:
  + The economic benefits from the project.
  + The economic costs of the project.
  + The net economic benefit, in which it can show if we benefited from it or not.
* **Schedule Feasibility:** As we go by the saying “Time is Gold,” time is a particularly important factor to consider. Would we be able to deliver functioning software that meets the customer’s requirements by the deadline? Bear in mind that a project will fail if not delivered by the required deadline! A detailed scheduling process and session should be considered.
* **Operational Feasibility:** When we are accepting the customer’s requirements for the software, we should be certain that we can deliver those requirements without any difficulties. Also, how easily can we maintain this type of technology after deployment? These are some questions to be considered and studied. In other terms, operational feasibility explains how efficiently we could use resources to give out a working product according to the customer’s requirements.
* **Legal Feasibility:** When we are creating any software, we should know that the user’s private information is not violated as well as how our software confines with international cyberlaws. If we sensed that a certain project violates some laws, we would immediately terminate it, this is because software should not be used in an unethical manner.
* **Technical Feasibility:** Technical feasibility is a study of whether our hardware and software resources are enough to develop and run such a project. This could also include how our staff would be capable of creating this project in a technical manner. In summary, we could describe technical feasibility as analyzing our resources to produce a project in a successful manner, these resources can include software, hardware, and our technical teams.

# **Software Behavioral Design Techniques**

In the process of designing a certain software, we can use certain diagrams and techniques that would allow us to thoroughly create this software. These diagrams are very good to have, in which they would guide us when creating the software and they would also reduce the percentage of uncertainty of not knowing how would the software react/work in the real world, under the hands of the general public.

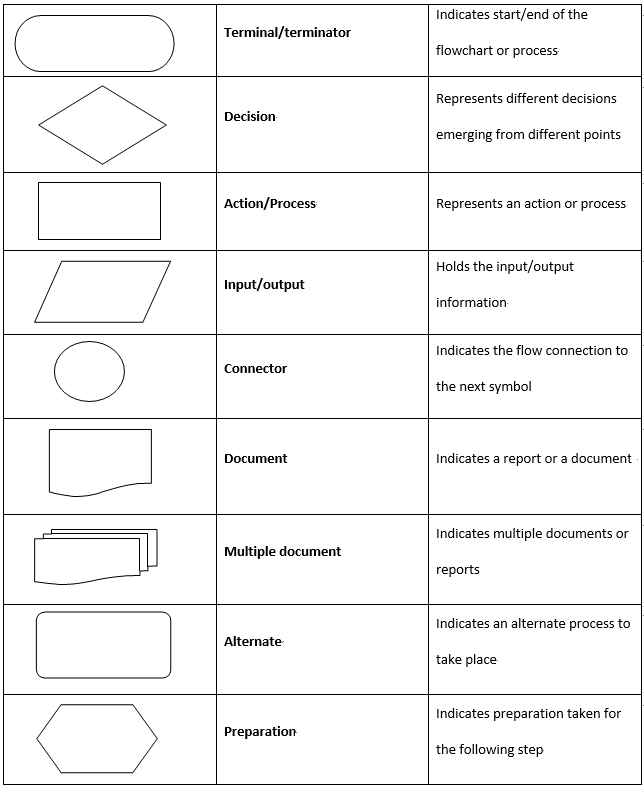
We have several diagrams and techniques that could be used, and I can assure you that every software that is created in this era has them all in their documentation. I would like to count a few:

* **Flowcharts.**
* **Pseudocodes.**
* **Finite and Extended State Machine.**
* **Data Flow Diagrams.**
* **Entity Relation Diagrams.**

In this report, I would like to talk in detail about two of these techniques, which are (Flowcharts and Entity Relation Diagrams). The reason why I chose these two techniques is that I felt that they would describe how the system would react and how different data members are related to each other.

## **Flowcharts**

To begin with, we should define what is flowchart and what is its purpose. A flowchart is a type of diagram that would show the sequential flow of steps for a certain algorithm, process, or workflow. Flowcharts contain a variety of shapes that would represent it, for example rectangles, ovals, diamonds which are then connected using arrows to show how they are connected to each other. Each shape has a different meaning, I would like to include an image that would illustrate those different shapes.



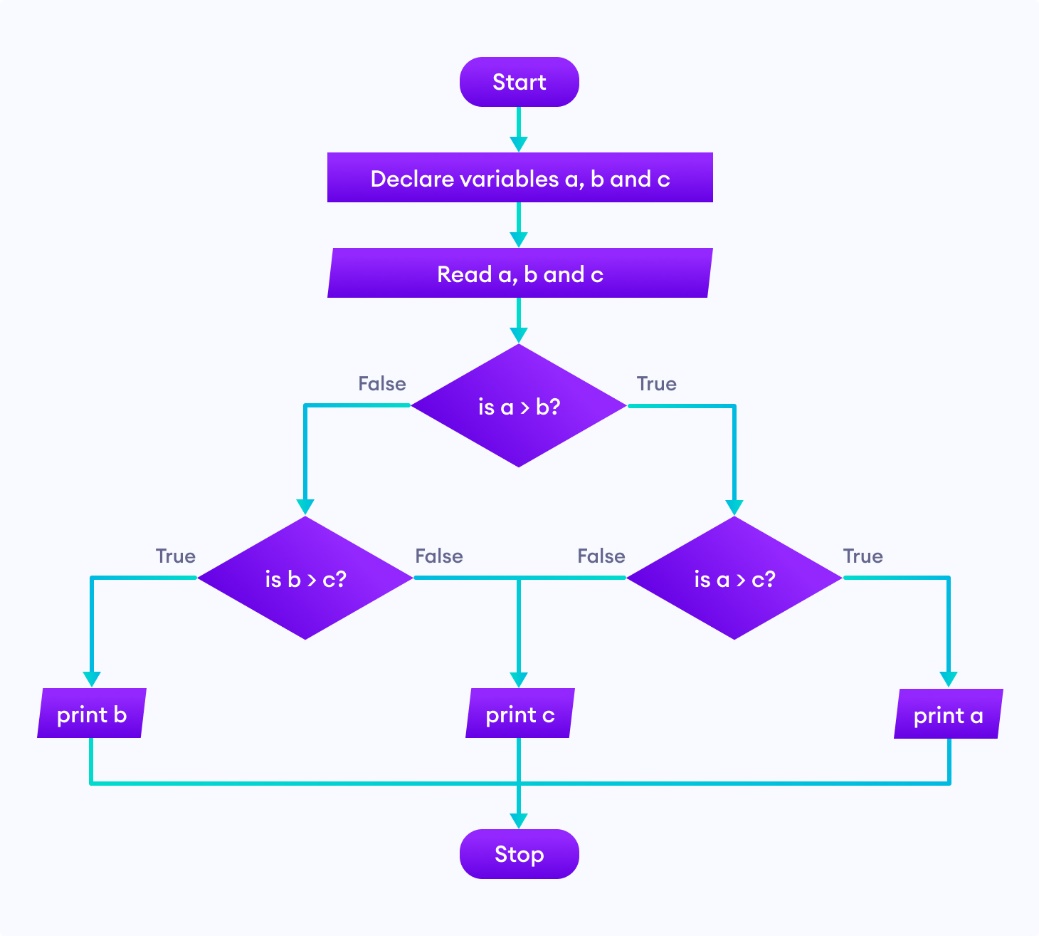
Flowcharts can be drawn to show a general outline of a process/algorithm or can be drawn in great detail that would show the tiniest steps that are involved. This of course depends on what the flowchart represent, is it a sophisticated process/algorithm?. I would like to talk about the general steps when creating a flowchart, which are:

* **Define the Purpose and Scope:** When creating a certain flowchart, we must fully know why we created this flowchart. Is it for a process? Is it for a certain algorithm that is used in our software? In which detail should we draw it? This step is an especially important step, in which when we have definite tasks to perform, this would make our lives much easier.
* **Identify Steps in Chronological Order:** Since this type of diagram is considered as a sequential type-of diagram, we should fully know the steps in order. It would not really make sense if we were in step #5, and we magically jumped to step #8.
* **Draw the Flowchart:** After setting out the pervious steps in a clear manner, we would actually start drawing the chart. This could be done as a simple sketch by hand, or we would also use specific software that could be used to illustrate it.
* **Confirm the Flowchart:** After drawing out the actual flowchart, we would go through it to ensure the integrity of the shapes and steps drawn. If we sensed that there are missing steps, we would redraw it and go through it from the beginning to the end another time.

What are some rules that must be followed to create a successful flowchart? We have four main rules that are present.

1. The flowchart opening statement must include a “start” keyword.
2. The flowchart ending statement must include an “end” keyword.
3. All symbols in the flowchart must be connected with an arrow line.
4. The decision symbol must not be associated with the arrow line.

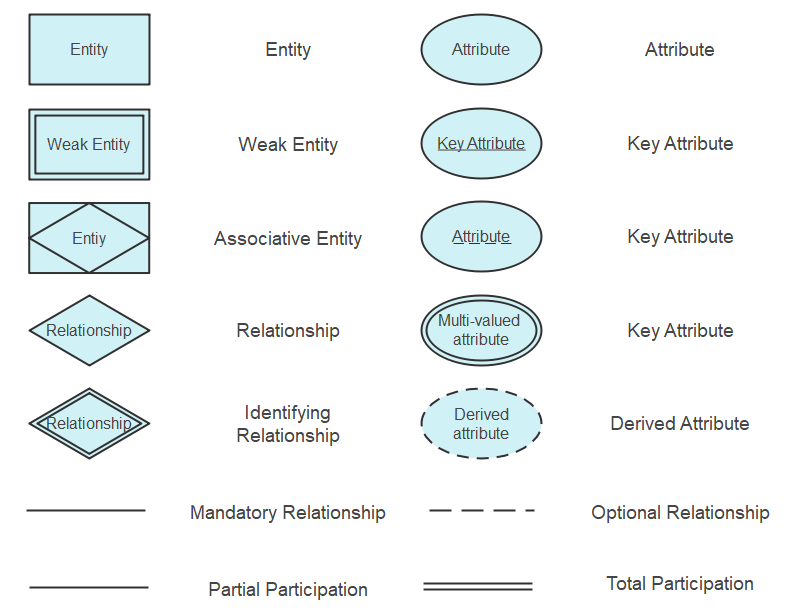
|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Easy to make and understand. | Although it is easy to make, it can be tricky when used to represent complex and large systems. |
| Flowcharts can provide a better understanding of the proposed system. | If any problem in it occurs, it would mean reproducing it from the start up. Which can be a time-consuming process. |
| Identifying mistakes can be easier. This can show any mistakes in the sense of logic in the proposed system. | When making flowcharts, there are no set of standards to determine the level of detail. |



In this graphic, it would show a flowchart of a program that would determine the largest number amongst three numbers.

## **Entity Relationship Diagrams**

An ERD (Entity Relationship Diagram) is a type of diagram that is used to represent and show how different tables in a database are connected to each other, the structure of the database, and the fields that are involved. Similar to flowcharts, ERD has a set of different symbols that help us define a database, I would like to provide a photo that justify this statement.



When creating an ERD, we have a set of components that we should consider as well as certain rules that would help us create a strong database model. I would like to define some terms that are used in ERDs.

* **Entity:** You can think about an entity as a thing, place, person, or object that is totally independent of each other, in which they would represent data components within a database.
* **Attributes:** An attribute is the set of properties that describe a certain entity. An attribute could include a variety of things, such as ID, gender, date-of-birth, first name, last name, etc..
* **Relationships:** Relationships would show how entities are related to each other as well as how they would communicate. In which it can be used to reduce redundant data and improve table structures. We have three types of relationships, which are:
  + **One-to-One:** In this kind of relationship, one entity in a table can relate only once to another table.
  + **One-to-Many:** In this kind of relationship, one record from a table would interact with one or more records from another table.
  + **Many-to-Many:** In this kind of relationship, many records in a table are associated with multiple records from another table.

**Diagram

Description automatically generated**

Any type of diagram would have certain rules that would be followed in order to create a successful and error-free diagram. In this case, we have some rules when creating an ERD.

* **Provide Proper Naming for Entities:** When naming an entity, we should avoid using vague naming and we should always try to use nouns. However, we can sometimes use adjectives to differentiate like-nouns, for example part-time employee and full-time employee.
* **Avoid Unnecessary Relationships:** Unnecessary relationships could create uncertainty and duplication of data. These false data could result in a false creation of queries.
* **Keep the Diagram Clear:** If the diagram is not easy to read and there is some overlapping in between the tables, it would be tricky to understand the database. Thus, causing a possibility for errors and uncertainty.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Generally, it is a simple model to use. We should only know what are the relationships that are present. Then we could draw the diagram easily. | It is fairly complicated when we want to represent any data manipulation. |
| Can be easily transformed into any other data model. These other data models can include hierarchical data model and network data model. | Like flowcharts, there is no standards to follow when creating the ERD. This could cause certain confusion amongst programmers that would read it. |
| Can provide better visual representation. This can mean a better representation of the relationships amongst entities. | ERD models provide a limited representation of relationships. |

Diagram

Description automatically generated

In this graphic, we would see how an ERD of a student enrollment system would be represented.

# **Software Behavioral Design Tools**

Different software tools are used to draw and illustrate those diagrams mentioned above. They are not only limited to these mentioned above, but endless types of diagrams can also be drawn. In this part of the report, I would like to talk about two famous tools that enterprises/companies use when they would like to create those important diagrams. I would like to talk about Draw.io and Microsoft Visio.

## **Draw.io**

Draw.io is an open-source tool that is used to create diagrams that are related to modelling and illustrating diagrams that are related to software creation. It is considered a budget friendly option, since it offers its services for free, of course there are some limitations and disadvantages when using such a tool. But to start things off as a mediocre developer/company that has limited budget, it is a great option.

Each program has its advantages and disadvantages, in the following table I would like to discuss some of them.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| It is cross-platform compatible. Meaning that it can be used from any machine using any operating system (Windows, MacOs, or Linux). | Compared to other more advanced tools, Draw.io is not a suitable option for complex and bigger projects, since it has only a limited number of shapes. |
| Real-time collaboration is one of the nice features that are included in Draw.io. Real-time collaboration is a feature that enables multiple user to access and work on a single project (for example an ERD). | After reading some reviews on the functionality of Draw.io, it has shown that some people had some problems uploading their project files from another diagraming software to Draw.io. This could cause a problem since there could be an integration gap. |
| Multiple integration technologies. Draw.io does not only use their technologies and restrict the user to it, however they can provide some flexibility. Some of these technologies include Google Drive, Confluence, and Jira. | Since it is a free tool, of course it cannot inhibit all of the advanced features that are covered by more sophisticated systems. These advanced features could include automation, data linking, and reporting. |

## **Microsoft Visio**

Microsoft Visio is one of the most popular diagramming software that is used on a more advanced level. By its name, it is endorsed by the company Microsoft, in which it would require a subscription. With such a complex system, Microsoft Visio requires a hefty subscription, some might question the offered price, but I am sure that it is a nice investment to have for complex software design.

In the following table, I would like to talk about the advantages and disadvantages of Microsoft Visio.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Microsoft Visio is used in a high-grade standard level; hence it would produce high-quality diagrams. | Since it is only Microsoft endorsed software, this means that we need Windows-running machines to be able to run the program. |
| Since it is from a Microsoft endorsement, integration with other existing Microsoft programs works flawlessly. | Compared to Draw.io, it has limited collaboration options. Thus, making it obligatory to stick with the available collaboration options, which are probably Microsoft products. |
| Since Microsoft Visio is used in more complex software diagrams, it has the ability to provide automated diagraming. Automated diagramming is important when we have dynamic variables. | Since it is a big and complex diagraming program, it is very hard to learn it all at once. This means that it has a steep learning curve and needs some training to use it. This training could cost the company time and money, which is not sometimes feasible in addition to the price of the license of the software. |

# **FSM vs EFSM**

We have two main state machines types, which are FSM (Finite State Machines) and EFSM (Extended Finite State Machines). Both are used to represent certain states that a software can cover, but there are some differences I would like to cover further on. We need to define what is a state machine diagram. It is a type of diagram that shows the sequence of events that an object goes through, in which it goes through a set of inputs and changes states depending on those given inputs. In other words, state machine diagrams model dynamic behavior.

## **FSM**

An FSM stands for a Finite State Machine is a type of model that is used to design and analyze how would a system react within a set of given inputs, in which it performs a “transition” from one state to another state. In FSM each transition has input Boolean conditions and output Boolean functions linked to it. In FSM only one single state in the system can be active.

## **EFSM**

On the other hand, we have something called an EFSM. EFSM stands for as an Extended Finite State Machine. Unlike FSM, EFSM can provide a more detailed representation of states. Since FSM uses Boolean conditions as an input, EFSM uses an "if statement" with a number of trigger conditions that can be used to express the transition.

# **Data-driven Software**

A software design approach known as data-driven software places a greater emphasis on data input than on pre-set rules created by programmers or by human interaction. Data-driven software approach differs from the traditional software design approaches in several aspects. One of these aspects is that there is always room for improvement, what I mean by this sentence is that it always gathers and processes data to be used for improving the software. Other software may only collect data at certain stages of their development. Also, one way that data-driven software is different than typical software development approaches is that they automatically adapt and reshape itself based on varying user behaviors, while other software design approaches need to be updated at some different intervals to be suitable for various user behavior.

Data-driven software is present in several different fields such as streaming companies, ride hailing applications, and prediction analytics systems. All of these industries need dynamic data to be fed into the system and as a result of that, it would provide a better and tailored user experience (Netflix for example).

Why do we use data-driven software in certain fields rather than the different methods that are available?

* It provides a better user experience. That is because it collects data from user interactions and behavior with the system, and therefore the system adjusts itself so that it is “tailored” for the user.
* A better performance is usually apparent in such systems. This means that these systems know how to allocate and use resources in a more linear and optimized way.
* Improved accuracy is also one of the benefits of using the data-driven methodology. Since the system relies on data, the chance of having errors, mistakes, and bugs is lower than the typical software methodologies.