Systems Analysis

System Life Cycle (or **SDLC** - **System Development Life Cycle**) is the process of developing information systems through investigation, analysis, design, implementation and maintenance. It is made up of the following number of steps:

1. Identification of the problem
2. Feasibility study and project selection
3. System Analysis (of old system)
4. System Design (of new system)
5. Programming and documentation
6. Implementation and changeover methods
7. Control and review
8. System maintenance

The above sequence of steps is called the **waterfall model**. Let us explain each step in detail. As an example, let us also consider a practical situation of a landing library that does not use any computers.

1. Identification of the Problem

This is the point when someone suggests the scrapping of a system and replacing it with a new one. Two reasons for this are:

1. The current system may be too slow.
2. Technological developments.

2. Feasibility Study and Project Selection

A **Systems Analyst** (i.e. the person responsible for the development of an information system) is hired to propose a project or a number of projects (from which one will be chosen). The systems analyst will:

1. Study the present system to see how it works
2. Propose a new system using computers, to replace the old system.
3. Give an estimate of how expensive the new system will be to build and to run. Give an estimate on the time it will take to build the new system.

4. All this will be presented in a feasibility report.

3. System Analysis

The customer who has ordered the feasibility study has chosen one solution and the systems analyst should proceed in the implementation of this project.

The systems analyst will carry out very detailed investigations in order to fully understand the current system and the proposed new system. She will see, for example:

- What is good about the current system
- What causes problems with the current system
- Which parts of the system are critical to the business/organisation?
- How the system can be made more efficient

The systems analyst has to learn all the details of the workings of the current system. This information can be gathered using these techniques:

- Interviewing staff at different levels of the organisation.
- Examining documents about the organisation.
- Questionnaires.
- Observation by spending time at various departments of the organisation.

4. System Design

During this stage the systems analyst:

- Must describe how the new system will work.
- She must decide on the following:
    - Screen layouts (user interfaces)
    - Menus
    - Inputs (e.g. bar code diagrams, OCR etc.)
    - Outputs (printer, screen, reports etc.)
    - Report formats
    - Hardware
    - Files
    - Programs to be bought off-the-shelf

    - Which programs should be tailor-made and what language/s are to be used?
- Prepare pseudocode or flowcharts or indicate general functions of a program
- Decide a conversion plan.
- Test plans

5. Programming and Documentation

The tailor-made programs are written and tested.

There are various types of testing. Some are mentioned hereunder. The tests, together with **test data**, are prepared at the design stage.

- Dry run testing: the programmer follows the code manually. A **trace table** will help (a trace table shows how variables are changing their values).
- Unit testing: Testing each module individually e.g. testing a method by itself.
- Integration testing: When all modules have been individually tested, integration testing ensures that they work together correctly.
- Black-box testing (functional testing): Testing the program without looking at the code.
- White-box testing (structural testing): The code is tested in such a way that each possible path of the program is executed during testing. This is a very thorough test.
- Alpha testing: Performed by testers within the organization that developed the software.
- Beta testing:
    - This involves giving the package to a number of potential users who agree to use the system and report any problems to the developers.
    - The product is corrected and can be sent out for further beta testing until the developer is confident enough in the product to put it on the market.

Documentation is an important part of software engineering. Types of documentation include:

- Requirements documentation: i.e. a description of what is required out of the system.
- Architecture/Design documentation: i.e. the plan of the new computer system.

- Technical documentation: i.e. documentation of the algorithms together with diagrams that show how the system works. This documentation is required for the system maintenance and upgrade.
- User documentation e.g. manuals (for the end user, system administrators and support staff), tutorials etc.
- Marketing: i.e. how to market the product and analysis of the market demand.
- Description of files used.
- Documentation of the tests performed on software.

Software should have the following characteristics:

- Usability: the users will find the system easy to use.
- Performance: the system is reliable and fast.
- Suitability: the system will be the best known for the problems it is trying to solve.
- Maintainability: it will be easy to make modifications and upgrades.

A **debugger** helps you to find where the errors in your program lie.

Some features of debuggers are:

- Breakpoints (a breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point).
- Stepping through the code
- Viewing the values of variables (tracing)
- See the values returned by functions
- Modifying the values of variables while you are executing and debugging a program
- Go back to a place in the program that has already been executed to see what the values of the variables at that stage were.

6. Implementation and Change-Over Methods

During this phase:

- All hardware, network, servers etc are put in place.
- Software is installed
- Users are trained for new system.
- New master files for the new system are created.

- Conversion from the old system to the new one takes place.

Conversion from the old to the new system (i.e. changeover) can take place as:

- Direct changeover.
    - The user stops the old system on one day and starts using the new system on the next day.
    - Advantage: fast and efficient with minimum duplication of work involved.
    - Disadvantage: if errors are found in the new system operations could be seriously disrupted.
- Parallel conversion.
    - The old system continues alongside the new system for a few weeks or months.
    - Advantages: (1) results of the new system will be checked against known results from the old system, (2) if difficulties occur work can continue by using the old system.
    - Disadvantage: duplication of work.
- Phased conversion.
    - It is used with a system that can be broken down into modules that can be implemented separately at different times.
    - Could be direct or parallel.
- Pilot conversion.
    - The new system will be used first by only a portion of the organisation.

7. Control and Review

After the system has been running for some time the system analyst checks the system. She would compare expected performance with actual performance. This acts as a check-up of the whole system e.g. security checks.

The systems analyst can talk with the users about their experience, expectations etc. She can also perform checks on the system herself. The more complex the system is the more controlling and reviewing the system is important.

8. System Maintenance

There are different types of maintenance:

- Perfective maintenance: the system is improved e.g. database response to queries made faster.
- Adaptive maintenance: changes are required e.g. new government legislation may mean that different methods of calculating tax are required.
- Corrective maintenance: i.e. correcting problems encountered in the new system.
- Predictive maintenance: use of techniques that help determine the condition of equipment in order to predict when maintenance should be performed. This approach offers cost savings over routine or time-based preventive maintenance, because tasks are performed only when warranted. At the same time, it prevents unexpected equipment failures.