Quick Computing Theory Notes (Part 2)

Systems Analysis (in a Nutshell)

# Systems Development Cycle

The **systems development cycle** is made up of the various **stages/phases** that have to be completed to create a **new modified computer system**.

It is a cycle as after a period of time, the system might need to be **modified/replaced** and the process has to be repeated.

# Step 0: Feasibility Study and Problem Definition

## Feasibility Study

The **feasibility study** is the **preliminary investigation** of a problem to decide whether a **solution is possible** and how the solution **may be done**.

It contains:

* **Context** of the problem
* **Evaluation/Simple analysis** of the problem
* **Ways** the problem can be solvable
* **Cost-benefit analysis** to determine whether the solution is affordable

## Terms of Reference

The analyst must:

* **Investigate** and **report** on the existing system
* **Specify objectives** for the system and **whether** they will be met by the new system
* **Recommend** the most suitable system to achieve the objectives
* **Prepare** a **cost-benefit analysis**
* **Prepare** a plan for **implementing the new system** within a short time scale.

## Factors for Feasibility

* **Technical** – is the technology feasible?
* **Economic** – is it economically feasible?
* **Social** – is the social effects likely to be damaging?
* **Availability** of **hardware/software**
* **Affordability** of **running** the solution
* **Time**
* **Skill** of workers
* **Effect** on customer

## Cost-benefit Analysis

### Costs

The **costs** of a new system may include:

* **Equipment costs** (computers and peripherals)
* **Installation costs**
* **Development costs** (of the system)
* **Personnel costs** (training, recruitment, salaries, etc.)
* **Operating costs** (consumables like disks, maintenance, etc.)

### Benefits

The **benefits** of a new system may include:

* **Savings** in personnel costs, operating costs, etc.
* **Extra sales revenue** due to better marketing information
* **Improved cash flow position** since invoices can be sent faster, etc.

## Why to computerise?

Some **manual systems** have characteristics that would be **more suited for computerisation**. These characteristics include:

* **Volume**
* **Requirement** for information to be **available** from several locations
* **Very accurate calculations**
* **Duplicated effort** involved (iteration)
* **Manual methods** are too **slow**
* **Data** has to be constantly **updated** and **accessible**

## Problem Statement

These problems in the current system which require the use of computerisation can be listed in the **problem statement**.

Some other reasons may be:

* **Transcription/Transposition errors** from human input
* **Layout** of organisation of data
* Etc.

If the solution is found to be feasible using computerisation, the **system development cycle** can start and the more advanced **systems analysis** process can take place.

# Systems Analysis

**Systems analysis** is the analysis of systems in businesses and organisations that help them run smoothly and efficiently. It is a **detailed look at the current system** and what the new system **will be required to do**. It is similar to the feasibility study but is **more detailed**.

**Systems Analysis:** the **detailed look** at **what the users require of the system** that the project is to implement. A **requirements specification is produced**, which forms the contract between the customer and the developer of the system.

A person who analyses systems is known as a **systems analyst**. They are usually employed by organisations and businesses to help them **improve their systems** and become **more efficient** or **profitable**.

## The System Development Life Cycle

|  |  |
| --- | --- |
| 1. Research | **Collecting information** on how the present system works |
| 1. Analysis | **Examining** how the present system works and identifying problems with it. |
| 1. Design | **Coming up with a new system** that will fix the problems of the current system. |
| 1. Development | **Creating the new system** from the design. |
| 1. Testing | **Checking** if the new system **works as expected** (doesn’t have any errors). |
| 1. Documentation | **Creating documents** that describe **how to use the new system** and **how it works** |
| 1. Implementation | **Replacing the present system** with the new system. |
| 1. Evaluation | **Checking** that the **new system meets all expectations**. |

# Step 1: Research

Before the systems analyst can make any recommendations about a **new system**, they first have to understand how the **present system** works.

As much information about the current system has to be **gathered** as possible. The techniques that can be used are:

## 1.1a Observation

The systems analyst **walks around** the organisation or business, watching **how things work** with their own eyes.

* Can gather **first-hand**, **unbiased** information
* People may **act differently** if they are aware they are being observed.

## 1.1b Collecting Documents

The systems analyst can collect examples of documents to **gain an understanding** of the **type and quantity of data that flows** through the business or organisation.

* If the documentation is **poor quality/insufficient**, collecting documents may not be very helpful.

## 1.2 Interviews

The systems analyst can **interview key people** within the current system to **find out how it works**.

* Gather a lot of **very detailed** information
* Interviews can take **a long time**, thus may not be feasible, especially if **a lot of people are involved** in the current system.

## 1.3 Questionnaires

The systems analyst can create a questionnaire to **gather information from large groups of people**.

* Can gather data from **many people**
* People **may not answer the questions seriously**, making the information **less reliable**.
* Information gathered is **limited to the questions asked** in the questionnaire by the systems analyst.

**NOTE:** If the question states that 3 methods of data collection for the current system are required, **state all four**, just that **observation** and **collecting documents** can be put into the same point.

# Step 2: Analysis

The systems analyst looks through the **information collected in Step 1** to **understand how the system works**, and to **try and identify problems** that need to be fixed.

## 2.1 Identifying Inputs, Outputs, and Processes

Every system has inputs and outputs, and the system analyst needs to identify the **data input and output to the present system**. This is because any new system that is designed will have to **deal with similar inputs and outputs** as the present system.

For similar reasons, the system analyst also has to **identify the processes** of the **current system**.

## 2.2 Identifying Problems

It is the job of the systems analyst to find out where the **problems in a system** are. If these problems are resolved, the system will work **more efficiently** and **smoothly**, and be more **profitable** for businesses.

## 2.3 Requirements Specification

The **requirements specification** is a **list of requirements** for the **new system**.  
The techniques for obtaining such requirements are:

* Interviewing
* Joint Application Design workshops
* Reviewing existing documents
* Analysing existing system
* Creating prototypes
* Observing current working practices

The new system designed must **meet these requirements**.

## 2.4 What software/hardware needed?

### Hardware

What **computers/network/servers**?  
Any **special input/output devices**? (e.g. barcode readers)

### Software

Are there any **existing off-the-shelf applications**?  
Does the software need to be **custom-made**?

## 2.5 Data Flow Diagrams

Data flow diagrams are diagrams that show **how data flows through a system**. These analysis tools show how the data is **input**, **output**, **stored**, and **processed** in a system.

# Step 3a: Design

## 3.1 Systems Flowcharts

The systems flowchart is a diagram used to **describe a complete data processing system**.

It describes it at an individual process level, and the flow of data through the operations is diagrammatically described, down to the level of the individual programs using the system requirements.

The details of the programs themselves are not included, as they are included with the **program documentation (Step 5)**.

It shows:

* The **tasks** to be **carried out** in the new system
* The **devices** to be used
* The **input/output media**
* The **files** used in the system

## 3.2 Other Design Tools

### Program Flowcharts

The program flowchart shows the operations involved in a computer program. It is part of the **permanent record** of a finished program for **maintenance (Step 7b)**.

### Pseudocode (covered in more detail on page 2)

Pseudocoding uses **control structures** and **keywords** like those in programming languages to describe a **program** or **system design**.

### Decision Table

It is a table that specifies the **actions taken when specific conditions arise**.

## 3.3 User Interfaces

### 3.3.1: Good UI Design

A good UI design takes into consideration:

* Who **uses** the system
* The **tasks performed** by the system
* The **environment** where the system is used
* What is **technologically feasible**
* **SAVE BUTTON !!!!!!!!!!**

### 3.3.2: Types of UI

Some types of UI include:

* **Command line interface** (CLI)
* **Menu** interface
* **Graphical user interface** (GUI)
* **Form** interface
* **Touchscreen** interface
* …and many more…

## 3.4 Data Inputs into a System

To get data into a system, data must first be **captured**, then **input** to a computer, either **manually** or using a **data capture device**.

### Some Data Capture and Input Methods

1. **Paper Forms**

Information is **written** into the forms, and **input** into the computer, either **manually** or using **machine-reading technology (OMR/OCR)**.

1. **Barcode Readers**

Barcode readers capture the **numeric code** represented by the barcode.

1. **Card Reader**

Card readers read data on the **magnetic strip/memory** on cards.

1. **Camera**

**Captures** **still** or **moving images** that can be **input** to the computer for processing.

## 3.5 Data Validation and Verification

### 3.5.1 Validation (covered in more detail on notes page 15)

**Data validation** checks whether the data input is **valid** or not.

The five types of **data validation checks** are:

1. **Presence** check - Is the data **present** within a field?
2. **Range** check - Is the data **within** the **specified range**?
3. **Length** check - Is the data **too short** or **too long**?
4. **Type** check - Is the data the right **type**?
5. **Format** check - Is the data the right **format**? (e.g. dates)

### 3.5.2 Verification

**Data verification** checks whether the data input is **correctly input** or not.

The two types of **data verification checks** are:

1. **Proof Reading**

A person compares the **original data** with the **data** in the computer.  
If mistakes are spotted, they can be **corrected** by the person.

* + **Quick** and **simple**
* **Doesn’t catch** every mistake

1. **Double Entry**

A person (preferably another person) re-enters the data into the system.  
If differences are spotted by the system, an **error** is generated and the person can **correct** the differences in the system.

* + Catches **almost every mistake**
  + **More time** and **effort** needed

## 3.6 Designing the System Processes

Any system has to **process** the data given. The **systems designer** has a number of things to consider:

### Designing Data and File Structures

A **data structure** is an **organised collection** of data. It is usually a **database** in which data will be **stored** as it is being **processed**.

When designing a **database**, the **systems designer** must consider:

* **Type** of data stored
* **Size** of data (length)
* **Field names** to use
* **How many records** to be stored

The **designer** also must consider what **backing storage device** or **medium** to store the data in:

* **Frequency** of accessing data
* **Speed** of accessing data
* **Size** of data files

## 3.7 Algorithms

To process the data, the **systems designer** must design the actual steps to be followed to process the data (**algorithms**).

## 3.8 Designing System Outputs

There are usually **two** types of output from a system that needs to be designed: **on-screen reports** and **printed reports**.

### On-screen reports

Designing an **on-screen report** is similar to designing an **on-screen form**.

When designing an **on-screen report**, the designer should:

* Show **all** necessary fields
* Have fields that are the **right size** for the data
* Have **easy-to-understand** instructions
* Make good use of **available screen area**
* Make good use of **colours** and **fonts** to make data clear

### Printed reports

Designing a printed report is similar to designing an **on-screen report**, just that it is **printed** on a piece of paper.

# Step 3b: Development

It is the process of **constructing** the **actual computer system** itself.  
It includes:

* Identifying the **modules** to be used and **specifying** them
* Identifying the **main data structure** within the programs
* Identifying the **main algorithms** to use as **pseudocode** or **structure diagrams**
* **Producing** the program and any other elements of the system

## 3.9 Software Development Cycle

The **software development cycle** is the sequence of steps taken to **produce working software**.

The stages are:

1. **Overall design** - identifies **what is needed** and **splits** it into **self-** **contained modules**
2. **Module design** - decides how **each module** performs its task
3. **Module production** - programs **each module** using a programming language.
4. **Module testing** - ensures that each module **works independently**
5. **Combining modules** to form the **complete system**
6. **Integration testing** - ensures that modules **work together**

## 3.10 Program design

It involves **drawing structure charts** and writing **detailed program specifications**.

## 3.11 Prototyping

It is the building of a **working model** of the system to **evaluate** it, **test** it, or **have** it **approved** before building the **final product**.

While some prototypes get **developed** into the final product, others are **discarded**.

# Step 4: Testing

**Testing** is the process of **detecting errors** in a system.

## 4.1 Test Plan, Test Data and Test Cases

### Test Plan

It is a plan containing **details** on **every single thing** to be tested.  
(e.g. does XXX work? / does this reject invalid data?)

It is **very detailed** and contain many **precisely specified tests**.

### Test Cases and Test Data

**Test data** are the data to be tested.  
**Test cases** are the **test data** and the **expected outcomes** from the test data.

## 4.2 Dry Run

A **dry run** (or desk checking) is a **manual check** through a program or system **step-by-step**. This is helpful in **locating errors** (especially run-time errors).

## 4.3 Unit and Integration Testing

**Unit test** – Each part of the system in **individually tested**.  
**Integration test -** All parts are **put together** and the **complete system** is tested.

## 4.4 Bottom-up and Top-down Testing

### Bottom-up Testing

* Components on the **lowest level** of the **hierarchy** are combined and tested first.
* The software is put together by including **successively higher-level** components.

### Top-down Testing

* The **skeleton** of the **complete system** is tested, where **individual modules** are replaced by ‘stubs’.
* These ‘stubs’ stand in for modules while they are **developed**. They may display a message stating that the module has been executed.
* In **subsequent tests**, the individual modules are included when they are **completed**.

## 4.5 White-box and Black-box Testing

### White-box Testing

**White-box testing** refers to testing that is done by the **programmers of the system** with the **knowledge** of the **underlying code** that runs the method. This helps the developers to test **every possible route** through the methods in the program.

### Black-box Testing

**Black-box testing** refers to testing that is done by **the system’s test engineers** whereby **no assumption is made** about how the code of the system works and the **test data** is obtained from an examination of the **requirements statement** of the system.

## 4.6 Developmental Testing

**Developmental testing** is the **repeated testing** of a **system** such that the results can be used for **further design and development**.

### Alpha Testing

**Alpha testing** is the issue of the software to a **restricted number of testers** within the **developer’s own company**. The alpha version may be **incomplete** and **have some faults**.

### Beta Testing

**Beta testing** is the issue of the software to a number of **privileged customers** in exchange for their **constructive comments**. The beta version are usually **similar to the finished product**. Beta testing takes place after the results of the alpha testing has been studied and **changes have been made**.

### Acceptance Testing

**Acceptance testing** is the testing carried out to **prove** to the **customer** that the system works correctly. It is carried out **after** the system is completed, and **ready to be handed over** to the customer.

## 4.6 Test Data

### Live Data

**Live data** is data that would normally be used in the current system.

### Normal, Abnormal, and Extreme Data Values

**Normal data** is data that would **normally be entered** into the system.

**Extreme data** is normal data, but at the **absolute limits** of the normal range.

**Abnormal data** is data that **should not normally be accepted** be accepted into the system, as the values are invalid.

## 4.7 Debugging, Errors, and Breakpoints

### Debugging

It is the **detection**, **location** and **correction** of faults/bugs that cause errors in a program. These errors are detected by **observing error messages** or by finding **unexpected results** in the test output.

### Errors

Errors are **faults** or **mistakes** in a computer program or system that causes it to produce the wrong results or not work. A **bug** is a fault in the program that **causes errors**. **Error messages** are generated by the computer to help the user **locate the likely source** of the errors.

Some types of errors include:

* **Execution errors -** errors detected **during program execution**, such as **division by 0 errors**, or **overflow errors**.
* **Compilation errors -** errors detected **during compilation**, such as  **syntax errors**.
* **Linking errors -** errors caused when a program is **linked to library routines**.
* **Syntax errors -** errors caused due to **incorrect program syntax**.
* **Logical errors -** **mistakes** in the **program design**, usually leading to program **displaying wrong results**.
* **Semantic errors -** errors caused by **violating rules** of the language.

### Breakpoints

It is a position within the program where the **program is halted** to aid in debugging. When the program is halted, the programmer can **investigate the values** of **variables, memory locations, and registers**. This helps the programmers to locate errors, particularly **run-time errors**.

## 4.8 System Testing

There are several ways to test the entire system:

**Functional Testing - ensuring all parts of the system works correctly**

**with test data.**

**Recovery Testing - ensuring that the system can cope and recover**

**from failures (power, hardware, etc.)**

**Performance Testing - tests whether the system can cope with a realistic**

**workload.**

# Step 5: Documentation

## 5.1 User Documentation

User documentation is intended to **help the** **users** of the system.

As the users are **non-technical people**, they do not need to know about how the system works, just **how to use it**.

User documentations may include:

* **Minimum** **hardware** and **software** required
* How to **install**, **start** and **stop** the system
* How to **use the features** of the system
* **Screenshots** showing **typical usage** of the system
* Example **inputs** and **outputs**
* Explanations to any **error message** shown
* **Troubleshooting guide**

## 5.2 Technical Documentation

Technical documentation is intended to **help the maintainers** of the system.

It provides information on **how the system works**.

Technical documentations may include details on:

* **Hardware** and **software** required
* **Data structures** used in the system
* **Expected inputs**
* **Validation checks**
* How **data** is **processed**
* **Data flow diagram**
* **System flowchart**

## 5.3 Systems Documentation

Systems documentation describes the results of **systems analysis**, what is **expected** of the system, the **overall design decisions**, the **test plan**, and the **test data** with the **expected results**.

## 5.4 Systems Specification

Systems specification is a **complete description** of the **whole system**, containing **data flow diagrams**, **system flowcharts**, **inputs**, **files**, **outputs**, and **processing**.

## 5.5 Program Documentation

Program documentation is the **complete description** of the **software intended for use** when **altering** or **adapting** the software, including the **purpose** of the software, **restrictions** on use of the software, **input** and **output** data, **flowcharts**, **program listings** and **notes** to assist in future modifications.

# Step 6: Implementation

The **implementation** of the new system occurs when the **old system is replaced**.

## 6.1 Direct changeover

The old system is **stopped immediately** and the new system **takes over**.

* + New system can be **started immediately**
  + If the new system fails, data is lost as there is **no back-up system**.

## 6.2 Parallel running

The new system is started but the **old system continues running for a short while in parallel** with the new system. After the new system is proven to work, the old system can stop operating.

* + If the new system fails, no data is lost as there is a **back-up system**.
  + The **outputs** of **both systems** can be **compared** to check that the new system is working correctly.
  + **Entering data** into two systems and **running** **both systems** takes up **more** **time** **and** **effort**.

## 6.3 Phased implementation

The old system is replaced by the new system **gradually, in phases**.

* + Allows users to **gradually** **get used** to the **new system**
  + Staff training can be done **in stages**
  + If the new system fails, data is lost as there is **no back-up system**.

## 6.4 Pilot running

The new system is **trialled** (pilot) in **one part of the business/organisation**.

* + Features can be **fully trialled**
  + Staff part of the pilot scheme can **train** **other staff**.
  + If the new system fails, data is lost as there is **no back-up system**, for the section of the business/organisation trialling the new system.

# Step 7a: Evaluation

The evaluation process **assesses the system** to see if:

* It does what it’s **supposed to do**
* It is **working well**
* Everyone is **happy** with it

## 7.1 What does an Evaluation look for?

When the systems analyst evaluates the new system, the following questions will be asked:

Is the system…

|  |  |
| --- | --- |
| * **Efficient?** | Does it **save time and resources**? Does it **operate quickly and with minimal waste**? |
| * **Easy to use?** | Can users use the system **with minimal training**? |
| * **Appropriate?** | Is it **suitable** and **meets the needs** of the business/organisation? |

## 7.2 How is a System Evaluated?

The systems analyst can use a number of techniques to evaluate the system:

### Checking against the Requirements Specification

The systems analyst **goes through the requirements** in the requirements specification **one-by-one** and checks whether the new system meets them.

### Checking the Users’ Responses

They can **obtain feedback** from the users of the new system, like in **Step 1**, through **questionnaires**, **interviews**, and **observation**.

## 7.3 Post-implementation Review

Once the system is up and running, a review needs to be performed to **confirm** that the new system is **fulfilling expectations**, and to identify any **weaknesses** or **modifications** that need to be made.

# Step 7b: Systems Maintenance

**Systems maintenance** involves:

* **Updating** the system to adapt it to **changing circumstances**, **legislation**, or **requirements**
* **Correcting** any errors that come to light
* **Documenting system updates** and **corrections**

There are several types of systems maintenance, including:

**Perfective Maintenance - making improvements, increasing ease of use**

**Adaptive Maintenance - take account of changes in business or legislation**

**over time**

**Corrective Maintenance - correct any errors that may have arisen**