Systems Analysis is, as the name states, the **analysis of systems**!

The systems that we are talking about are the **systems within organisations and businesses** - systems of communication, financial systems, manufacturing systems, etc. - basically the **systems** that make the organisation or business **work**.

A person who analyses systems is known as a **Systems Analyst**.

Often systems analysts are **employed** by organisations of businesses to help them **improve their systems** and so become more **efficient**, and for businesses, more **profitable**.

A systems analyst would generally perform the following steps in the order shown...

1. **Research/System Study**
   Collecting information about the present system works.
2. **Analysis**
   Examining out how the present system works and identifying problems with it
3. **Feasibility Study**
   A study that would tell whether it is okay to go with the designing of the sytem
4. **Design**
   Coming up with a new system that will fix the present systems problems
   Creating the new system from the design.
5. **Testing**
   Checking if the newly created system works as expected
6. **Documentation**
   Creating documents that describe how to use the new systems, and how it works
7. **Implementation**
   Replacing the present system with the new system
8. **Evaluation**
   Checking that the new system meets all expectations

Analysing the Present System

Having collected as much information about the present system as possible, the systems analyst now **looks though it all** to understand **how the system works**, and to try and **identify problems** that need to be fixed...

Analysis

Identifying the Inputs, Outputs and Processes

Every system has **inputs** and **outputs** and the systems analyst needs to identify the data input to the present system, and the data output.

This is because any **new system** that is designed will have to deal with **similar inputs and outputs** as the present system.

For example, the **payroll system** in a business might have the following inputs and outputs...
Identifying the inputs, outputs and processes helps the Systems Analyst really understand how a system works:

- *What goes in?*
- *What happens inside?*
- *What comes out?*

Any **new system** that is created will need to take in the **same input data** (the number of hours worked by employees), and will have to produce the **same three outputs**.
For similar reasons, the systems analyst also has to understand **how the present system works** (the processes – who does what and when)...
This is important because some parts of the present system may work very well, and it would be a waste of time and effort to replace them.

Researching the Present System

Before the systems analyst can make any recommendations about a new system, they first have to understand **how** the present system **works.**
Gathering / Collecting Information
As much **information** about the present system needs to be gathered as possible.

Observation

This involves the systems analyst walking around the organisation or business, watching how things work with **his/her own eyes**.

Observation allows the systems analyst to gather **first-hand**, **unbiased**information.

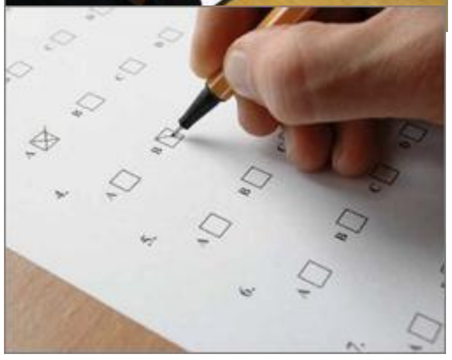
The downside to observation is that often people **won't work the way they normally do** if they know they are being watched.

Interviews

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

Interviews allow lots of **very detailed information** to be gathered, but they take a **long time** to do, so are not possible if large groups of people are involved.

Questionnaires



With **large groups** of people, a questionnaire is a **quick** and simple way to gather information.

However the information gathered is **limited** by the questions set by the systems analyst (people could have a lot of useful information in their heads, but if the questionnaire doesn't ask the **right questions**, they will not be able to pass it on)

Also many people do not take the **time** to fill in questionnaires **seriously**.

Collecting Documents

Most businesses and organisations use documents to record **information**, or to **communicate** information (**forms** get filled in and passed to other offices, etc.)

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

Identifying Problems

No system is perfect and it is the job of the systems analyst to try and **identify where the problems in a system are**.

If these problems can be **fixed**, the system will work more smoothly, be more **efficient** and, in the case of a business, be more **profitable**.

New System Requirements Specification

Now the problems with present system are understood, the system analyst can begin to plan how the new system will **fix those problems**.

The systems analyst **specifies** a **list of requirements** for the new system ('requirements' simply means **targets** or aims).

This list is usually called the **Requirements Specification**.

What Hardware and Software Will Be Required?

The systems analysts will now need to decide what hardware and software will be required for the new system...

Hardware

- How many **computers**?
- What type of **network**?
- How many **servers**?
- Any special **input devices**? (e.g. barcode readers)
- Any special **output devices**?

Software

- Is ready-made, **off-the-shelf** software available?
- Is **custom-written** software required?

Off-the-shelf *software is software that is created for use by a large range of customers - it tends to be quite* ***general-purpose****.*
Custom-written *software is designed and written* ***specifically for one customer****.*

Off-the-shelf *software:*

- ***Cheaper***

- *More **reliable** (because most problems will have been found by one of the many users)*
- *Has **lots of support** and help available (because lots of other people are using it)*

Custom-written *software:*

- *Very **expensive***
- *Provides **exactly** what the customer **needs** (a 'perfect fit')*
- *Only has one user, so **little help is available***
  - Designing a New System

Using the list of requirements, the systems analyst now has to **design** the **new system**.

In most cases the new system will be **computer-based**. The ease with which computers can communicate and process data means that are usually the best tool for the job.

Feasibility Study/ Report

A feasibility report is prepared after the detailed study of the system that is going to be developed. This report will identify the main issues that need to be discussed

- Can the company afford financially to get the system developed?
- Does the company have the manpower to do all this?
- Is the building environment right for the system to take place?

Design

Designing the System Inputs

To get data into a system is a two-part process:

- Data must first be '**captured**' (collected in a way that then makes it easy to input)
- Data must be **input** into the computer

The systems analyst will select a data capture method and data input method that best suit the requirements of the new system.
*Sometimes the two steps of **data capture** and **data input** are performed at the **same time**.*

*For example a **barcode reader** captures the data (the numeric code on the barcode) and inputs it to a computer in one go.*

Choosing the Best Data Capture and Data Input Methods for the System
Collecting data into a form that is ready for input to a computer system can be done in many ways...

- **Paper Forms**

Form can be a simple ones with **spaces for numbers and text** to be written in. The data form this form would then be **typed** into the computer

Forms can also be **machine-readable**, such as **OMR** forms

- **Barcode Reader**

Barcode readers capture the **numeric code** that the barcode represents.

Typically used with **POS** systems and also **stock-control** systems

- **Card Reader**

Many cards contain data stored on a **magnetic strip** or in a small bit of **memory** (smart cards) which can be captured with a **card reader**

Used in systems such as **EFTPOS**

- **Camera**

Capture **still or moving images** which can then be input to a computer for processing
Designing On-Screen Forms for Data Input

Much of the data that enters computer systems needs to **typed in**. A **well-designed on-screen form** can make this task **easier** and **quicker**.

On-screen forms should...

- Have all of the necessary **fields**
- Have **obvious** places for user input (boxes, use of colour, etc.)
- Use appropriate **controls** (see right) for each field
- Have text box controls that are the **right size** for the data
- Have easy-to-understand **instructions** (if needed)
- Make good use of the screen **area** available

This is an example of a well-designed on-screen form used to enter details of an employee…
As data is entered into the form, it needs to be checked for **accuracy**. Two techniques help us do this: **validation** and **verification**...

Data Validation Techniques

When data is input to a computer, it is a good idea for the computer to check that the data is **sensible** (no dates of birth in the future, etc.)

Checks like this are called **validation checks** (is the data **valid**?)

Different validation checks can be used on different fields, depending on the **type** of data being entered...

Presence Check

- Is data actually **present** in a field, or has it been missed out?

Range Check

- Is the data value **within a set range**?
  (E.g. an exam mark should be between 0% and 100%, a month should be between 1 and 12)

Length Check

- Is an item of text **too short or too long**?

Type Check

- Is the data the correct **type**?
  (E.g. the letter 'A' should not be allowed in a numeric field)

Format Check

- Is the data in the correct **format**?
  (E.g. a date of birth should be entered as dd/mm/yyyy)
- *If one of the validation checks **fails** (because the data entered is **invalid**) the computer should show a nice, **friendly error message** such as...*

  "You have forgotten to enter a name"

Data Verification Techniques

Data **validation** only checks whether the data entered is **sensible -** it does not mean that the data is the **right data**.

For example, if you are entering a date of birth and you mis-type it…
Correct date of birth: **12/11/1982**
Date of birth entered: **12/11/1928**
. . . you would not see an error, since 12/11/1928 is a **valid date** of birth.

To check that data is the **correct** value, we use a system called **data verification**.

There are two methods of data verification...

1. ## Proof Reading

After the data has been entered a **person compares the original data with the data in the computer** (either on the screen or using a print-out).

If mistakes are spotted they can be corrected by the **person**.

Proof-reading is **quick** and **simple**, but **doesn't catch every mistake**.

2. ## Double-Entry

The data is **entered into the computer twice** (preferably by two different people).

The computer compares the two sets of data to see if they match. If not it generates an **error** and a **person** will need to **correct** the mistake.

Double-entry takes more **time** and **effort**, but it **catches almost every mistake**.

Designing the System Processes/Database designing

Any system has to process the data it is given. The system designer has a number of things to consider...
Designing Data and File Structures
A data structure is an **organised collection of data**. Most commonly, this will be some sort of **database** in which data will be stored as it is being processed.

When designing a database, the system designer needs to consider:

- The **type** of data being stored (numbers, text, dates, etc.)
- The **size** of the data (how long is a typical name, etc.)
- The **field names** to use

- **How many records** will need to be stored

The designer also need to consider which **backing storage device and media** will be suitable to store the data:

- How **often** will the data need to be accessed
- How **quickly** the data needs to be accessed
- How **large** will the data files be

Designing the System Outputs

There are usually two types of output from a system that need to be designed:
On-screen reports (information displayed on the monitor)
Printed reports (hard-copy to be mailed, filed, etc.)

Designing On-Screen Reports

- Designing an on-screen report is similar to designing an on-screen form (see above). There are a number of things that the designer should consider.

  On-screen reports should...
  - Show all of the necessary **fields**
  - Have fields that are the **right size** for the data
  - Have easy-to-understand **instructions** (if needed)
  - Make good use of the screen **area** available
  - Make good use of **colours** and **fonts** to make the data clear

Reports can include:

- ***Text***
- ***Images***
- ***Bar charts***
- ***Pie charts***
- ***Animations***
- ***Video***

Designing Printed Reports

- Designing a printed report is just like designing an on-screen report (see above), except that the report needs to fit a piece of **printer paper**, rather than the screen. The report might also include page numbers, a header / footer, etc

  This is an example of a well-designed printed report used to show details of an employee…

Testing the New System
Once the system has been created, it needs to be thoroughly tested.

A **test** plan is usually written whilst the system is being developed. The test plan will contain details of every single thing that needs to be tested.

Testing

For example:

- Does the system **open and close** properly?
- Can data be **entered**?
- Can data be **saved**?
- Can reports be **printed**?
- When you do something **wrong**, does an **error message** appear?
- Is **invalid data rejected**? E.g. if you are not allowed to enter an amount above £1,000 on the system then a value of 1,001 should not be accepted (i.e. does the **validation** work?)

Test plans are very detailed, and contain many tests. Each test is specified very precisely.

A typical test would contain:
Details of **what** is being tested
The **test data** to use
What is **expected to happen** when the test is performed
Selecting Test Data
When choosing what data to use to test a system, you need to think about why we are testing the system: to see if it works, and to check it doesn't break.

To do this, we use three types of test data...

Normal Data Values

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

The system should accept it, process it, and we can then check the results that are output to make sure they are correct.
*E.g. In a system that was designed to accept and process test marks (**percentages**), then **normal** test values would include:*
10
25
63
89

Extreme Data Values

Extreme value are still **normal** data.
However, the values are chosen to be at the absolute **limits of the normal range**.
Extreme values are used in testing to make sure that **all normal values** will be accepted and processed correctly.
*E.g. In a system that was designed to accept and process test marks (**percentages**), then extreme test values would be:*
*0 (**lowest** possible value)*
*100 (**highest** possible value)*

Abnormal Data Values

This is data that should **not normally be accepted** by the system - the values are **invalid**.

The system should **reject** any abnormal values.

Abnormal values are used in testing to make sure that invalid data does not **break** the system.
*E.g. In a system that was designed to accept and process test marks (percentages), then **abnormal** test values would include:*
-1
101
200
-55

Documenting the New System
There are two types of documentation that should be produced when creating a new system:
User documentation and **Technical** documentation

Documentation

User Documentation

The user documentation is intended to help the **users** of the system.

The users are usually **non-technical** people, who don't need to know how the system works. They just need to know **how to use it**.

User documentation usually includes:
- List of **minimum hardware and software** required to use the system
- How to **install** the system
- How to **start / stop** the system
- How to **use the features** of the system
- **Screenshots** showing the system in typical use
- Example **inputs and outputs**
- Explanations of any **error messages** that might be shown
- A **troubleshooting guide**

If you've ever purchased any computer software (e.g. a game), then you will have seen a user guide - it's the little booklet that comes in the CD case (that you didn't read!)

If you buy a car, it comes with a user guide that explains how to start it, unlock the doors, fill it with fuel, etc. (nobody reads those either!)

Technical Documentation

The technical documentation is intended to help the **maintainers** of the system (the people who need to keep the system running smoothly, fix problems, etc.)

The maintainers are usually **technical** people, who need to know exactly **how the system works**.

Technical documentation usually includes:
- Details of the **hardware and software** required for the system
- Details of **data structures** (data types, field names, etc.)
- Details of **expected inputs**
- Details of **validation checks**
- Details of **how data is processed**
- **Diagrams** showing how data moves through the system
- **Flowcharts** describing how the system works

If you buy a car, you wouldn't normally want the technical documentation for it.

Your mechanic would be the person who would need the technical documents. If the car needed servicing, or fixing, the mechanic would look in the document to understood how the various systems in the car worked.

Implementing the New System

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

There are a new of ways of implementing a new system...

Implementation

Direct Changeover

The **old** system is **stopped completely**, and the **new** system is **started**. All of the data that used to be input into the old system, now goes into the new one.

This is has its **advantages**...

- Takes the **minimal time and effort**
- The **new** system is up and running **immediately**
- But there are also **disadvantages**...
- If the new system fails, there is **no back-up system**, so data can be lost

Sometimes a direct changeover is the only way to implement a new system.

E.g. an aircraft's auto-pilot system can't have a new and old version running side-by-side, arguing about what to do!

Parallel Running

The **new** system is **started**, but the **old** system is **kept running in parallel** (side-by-side) for a while. All of the data that is input into the old system, is **also** input into the new one.

Eventually, the old system will be stopped, but only when the new system has been proven to work.

This is has its **advantages**...

- If anything goes wrong with the new system, the **old system will act as a back-up**.
- The **outputs** from the old and new systems can be **compared** to check that the new system is running correctly
- But there are also **disadvantages**...
- Entering data into two systems, and running two systems together, takes a **lot of extra time and effort**

Phased Implementation

The new system is introduced in **phases**, gradually replacing **parts** of the old system until eventually, the new system has taken over.

This is has its **advantages**...

- Allows users to **gradually get used to** the new system
- Staff training can be done in **stages**

But there are also **disadvantages**...

- If a part of the new system fails, there is **no back-up system**, so data can be lost
- Evaluating the New System
- Once the new system has bee implemented and is in full use, the system should be evaluated (this means that we take a long, critical look at it).

  The purpose of an evaluation is to assess the system to see if it does what it was **supposed** to do, that it is **working well**, and that everyone is **happy** with it.

Evaluation

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 operate quickly, smoothly and with minimal waste?
Is the system saving time, and resources?
...**easy to use**?

Are all of the system's users able to use the system easily and effectively?
Can new staff understand and use the system with minimal training?
...**appropriate**?
Is the system suitable for the particular business / organisation?
Does the system actually meet the needs of the business / organisation?
But how can we find the answers to these questions?
How is a System Evaluated?
The systems analyst will use a number of techniques to evaluate the system...
Check against the

Requirements Specification

If you remember, earlier on in the Systems Analysis, the old system was **analysed**, and a **checklist of targets** was drawn up for the new system.

This list was called the Requirements Specification.

The systems analyst will use this document to check the new system. Going through the **requirements** one-by-one the analyst will check if they **have been met**

Check the Users' Responses
It is essential to get feedback from the **users** of the system...

- Do they like it?
- Does it make their work easier?
- What, if anything, could be improved?

The systems analyst can get this feedback in the same way they collected information about the original system...
Questionnaires
Interviews
Observations
What Happens Next?
The outcome of the evaluation will be to identify any **limitations** or **problems** with the new system.

The system analyst will then need to begin the task of system analysis from the **beginning**, but this time analysing the **new** system, and then designing, testing and implementing **improvements**.

Future Developments

Thus the whole process repeats...
The fact that the process of Systems Analysis is often ***repeated*** *over and over (constantly building upon and improving systems) means that it is often referred to as a* ***cyclic*** *(repeating) process.*

The stages of Systems analysis are often shown in a circle