Systems Life Cycle

- Large systems development projects involves many people working for many months.
- Development process has to be planned well to ensure that the project is delivered on time and to the required specification.
- System life cycle development process is very methodical and sequential.
- Each stage is composed of well defined activities and responsibilities and has to be completed before the next stage begins.

Classical Waterfall Model

- Software life cycle is also referred to as the “Waterfall Model”
- The Waterfall Model usually includes the following processes:
  - Problem definition
  - Feasibility study
  - Requirements analysis
  - Design
  - Programming and module testing
  - Integration and system testing
  - Delivery
  - Maintenance
Waterfall Model

- One stage feeds its output or deliverable into the next stage
- The deliverables from every stage are clearly defined
- All parties involved (end-users, management and developers) have to approve the deliverables before the next stage can be started
Problems

- Difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.
- Inflexible partitioning of the project
- Only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
Feasibility Study (TELOS)

- **Main aim**: understand the problem and determine as much as possible all the issues involved should the problem be tackled
- **Technical feasibility**: investigate whether the technology exists to implement the proposed system, or whether it is a practical proposition
- **Economic feasibility**: establish if proposed system is cost-effective – if benefits do not outweigh costs, it’s not worth going ahead.

---

Feasibility Study (TELOS) [cont]

- **Legal feasibility**: determine if there is any conflict between the proposed system and legal requirements – e.g. the Data Protection Act
- **Operational feasibility**: determines if the current work practices and procedures are adequately supported by the new system: how change affects the working life of workers
- **Schedule feasibility**: looks at how long the system will take to develop, or whether it can be done in a desired time-frame
- **Deliverable** from this stage is a feasibility report produced by the system analyst. If the project is to proceed then a project plan and budget estimate for the other stages of development will be produced.
Systems Development

Requirements Specification and Analysis

Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.
What is a requirement?

- May range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- Requirements may serve a dual function
  - May be the basis for a bid for a contract (interpretation)
  - May be the basis for the contract itself (detail)
  - Both these statements may be called requirements.

Functional Requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
  - functional user requirements may be high-level statements of what the system should do
  - functional system requirements should describe the system services in detail.
Example: Functional Requirements

- Some functional requirements of a university library system
  - to search for a library item by specifying a key word
  - to issue a library item to a library member
  - to reserve a library item on line

Non-Functional Requirements

- These define system properties and constraints
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.
Non-Functional classification

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

ATM Example: Functional and Non-Functional Requirements

- The functional requirements of the ATM:
  - authorization process
  - transaction (withdrawal process)
- The non-functional requirements
  - The ATM network has to be available 24 hours a day.
  - Each bank may be processing transactions from several ATMs at the same time.
  - The ATM must be able to use several data formats according to the data formats that are provided by the database of different banks.
Requirements completeness and consistency

- In principle, requirements should be both *complete* and *consistent*.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document.

Problems of requirements analysis

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.
Requirements Analysis Process

During this phase a more detailed investigation into the current system and its requirements are made.

This phase can be divided into 3 sections:

- **Determining requirements**: Persons responsible of this phase have to gather information about the status of the existing system and the requirements of the new one. This entails defining a set of questions, deciding on which sources of information to use and determine the most appropriate information-gathering technique to use.

- **Analyzing these requirements**: The collected data will be used to gain a clear understanding of the status of the present system and the requirements of the new system. In order for this process to be effective the raw data needs to be presented in an appropriate format.

- **Produce a report**: When the analysis has been undertaken, a report is presented to the manager responsible for approving the next stage of the development.

The Analyst

- Someone with the ability to capture user and system needs at different levels
- Someone who can envisage a system from different aspects (points of view)
- Someone who can communicate ideas on a non-technical basis
- Someone who can save (or cost) an organization a lot of effort and money
Determining Requirements

<table>
<thead>
<tr>
<th>What questions need to be asked?</th>
<th>What sources of information can be used?</th>
<th>What information gathering techniques can be used?</th>
</tr>
</thead>
<tbody>
<tr>
<td>• What function does the current system perform?</td>
<td>• External sources:</td>
<td>• Structured interviews with individuals, i.e. questions determined in advance</td>
</tr>
<tr>
<td>• What are the shortcomings in the current system that the new system will remove?</td>
<td>• Customers/clients</td>
<td>• Unstructured interviews with individuals, i.e. open-ended discussions</td>
</tr>
<tr>
<td>• What data will be required for the new system?</td>
<td>• Suppliers</td>
<td>• Observation of existing practice</td>
</tr>
<tr>
<td>• What are the sources of the data?</td>
<td>• Government agencies</td>
<td>• Reading of documentation, journals, manuals</td>
</tr>
<tr>
<td>• How will the data be collected and inputted?</td>
<td>• Consultants</td>
<td>• Questionnaires to users</td>
</tr>
<tr>
<td>• What outputs will be required by the new system?</td>
<td>• Competitors</td>
<td>• Examination of documentation related to current system</td>
</tr>
<tr>
<td>• How will the old system interact with the new one?</td>
<td>• Direct users</td>
<td>• Visits to other organisations</td>
</tr>
</tbody>
</table>

Scenarios/Use-Cases

- **Scenarios**: real-life examples of how a system can be used. They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

- **Use-cases**
  - Use-cases are a scenario-based technique which identify the actors in an interaction and which describe the interaction itself.
  - A set of use cases should describe all possible interactions with the system.
Example: ATM Scenario

- **Scenario**: A successful withdrawal attempt at an automated teller machine (ATM).
  - John Smith presses the Withdraw Funds button
  - The ATM displays the preset withdrawal amounts (Lm25, Lm30, etc)
  - John chooses the option to specify the amount of the withdrawal
  - The ATM displays an input field for the withdrawal amount
  - John indicates that he wishes to withdraw Lm50
  - The ATM displays a list of John’s accounts, a checking and two savings accounts
  - John chooses his checking account
  - The ATM verifies that the amount may be withdrawn from his account
  - The ATM verifies that there is at least Lm50 available to be disbursed from the machine
  - The ATM debits John’s account by Lm50
  - The ATM disburses Lm50 in cash
  - The ATM displays the “Do you wish to print a receipt?” options
  - John indicates “Yes”
  - The ATM prints the receipt
Analysing Requirements

- There are several analytical techniques that can be applied to achieve this.

Data Flow Diagrams

DFDs represent diagrammatically the relationship between different systems in the organisation and how data moves around in these systems. They contain four main components:

- Entities (e.g. customers)
- Processes (e.g. computing total)
- Data Flow (i.e. the origin, destination and direction of data transfer)
- Data Store (any location where data is held e.g. customer data file)

Data Flow Diagrams (DFD)

- Shows how data moves through a system and what data stores are used.
- It does not specify what type of data is used and how the data is stored.
DFD Symbols

- **External Entity**: Data source or data destination; e.g., people who generate data such as a customer order, or receive information such as an invoice.

- **Process**: An operation performed on the data. The top section of the box is used to label the process. The middle part is used to give a brief explanation.

- **Data Store**: Represents data store such as a file held on disk or a batch of documents.

- **Data Flow**: Represents the movement between entities, processes, and data stores. It should be labelled to describe what data is involved.

DFD: Different levels of abstraction

- **View the system at different levels of details**
  - **Level 0 (Context diagram)**
    - Comprises a single process, all agents, and input/output data flows
    - Purpose: Identify and examine the external interfaces
  - **Guidelines for drawing Level 0**
    - Look for entities that give data to the system
    - Look for entities that accept data from the system
    - Identify the data elements at an abstract level
    - Draw the diagram