Notes

Contents

1 What is a System?

System is an interconnected components that work together to achieve a goal

- Any System:
  - Designed to perform specific and various goals and objectives for the organizations or business.
  - consists of number of processes that convert the input to output
  - Have data and information to maintain.

1.1 Examples

An Organization may also be described as a system where all staff interact with each other to become a functional unit that could communicate with their clients to make a complete business system.

1.2 System Components

- Resources (Can be shared)
  - Financial Resources
  - Hardware
  - Human Resources (HR)
  - Software
  - Time
  - Fund, Money
- Procedures/Rules
  - Defined by the organization
  - shall be followed as standard
  - Ensure full compliance with legislative requirements
- Processes/Functions
  - Operations components of the System
  - Could have feedback to improve the processes

1.3 System Characteristics

- Input
- Output
- Process
- Feedback
  - Compares the Systems output and the system's goals
- Boundaries
- Integration

- Environment

2 What is an Information system?

- Collection of interrelated components:
  - Collect
  - Process
  - Store
  - Provide output to complete a business task.

2.1 Examples

- Course registration
- Online Banking
- Online Order

3 System Analysis and Design

The method used by Software development companies to create information systems/Software

- The method used by Software development companies to:
  - Create information systems/software
  - Maintain SW information systems/software
    - To perform basic business functions
- Transforms a project idea into documentation and application

3.1 System Analysis

- Understand the goals and strategies of the business to investigate the problem.
- Investigate the information system requirement then decide the requirements to serve the goals and strategies
- No Code

3.2 System Design

- A conceptual solution that fulfills the requirements
- **NOT AN IMPLEMENTATION**

4 Who is the Systems Analyst?

The systems analyst plays a key role in *information systems* development projects. The systems analyst works closely with all project team members so that the team develops the right system in an

effective way. Systems analysts must understand how to apply technology to solve business problems. In addition, systems analysts may serve as *change agents* who identify the organizational *improvements* needed, **design** systems to implement those changes, and train and motivate others to use the systems.

- Uses system analysis and design techniques to solve business problems using IT
- Works closely with all project team members
- work in all phases.
- use project management methods.
- Mainly business-oriented.
- The primary Objective of the System analyst is to create value for the organization, not a wonderful system.
- Interact with:
  - customer
  - Technical team
  - Business team
  - vendors
  - consultants
  - Needed skills/attributes:
    - Technical knowledge
    - business knowledge
    - analytical
    - interpersonal
    - management
    - ethical
    - accept challenges
    - problem solving skills
    - effective written communication
    - effective oral communication
    - good and active listener

4.1 System Analyst Skills

- The skills can be broken down into six major categories
  - Technical
    - To understand the organization's existing technical environment
    - To understand the new technology
    - To find a way to fit both organization's existing technical environment and new technology into an integrated technical solution
  - Business
    - required to understand how IT can be applied to business situations
    - Ensures that IT delivers real business value
  - Analytical

    - Problems solvers at both project and organizational level
  - Interpersonal
    - Effective communication
      - one-on-one with users, business managers, and programmers.
      - Must be a good listener
  - Management
    - Ability to manage people with whom they work, manage the pressure and risks associated with unclear situations.
  - Ethical

4.2 Systems Analyst Roles

- System Analyst
  - Focuses on IS issues
  - Develop new ideas and suggestions for ways that IT can support and improve business
  - Helps design new business processes
  - Designs new information systems
  - Ensures all IS standards are maintained.
  - view the system as a whole and bridge the gap between IT and business
- Business Analyst
  - Focuses on business issues
  - identify the business value that the system will create.
  - Develops ideas for improving the business processes
  - Helps design new business processes and policies
- Consultant
  - bring with them a fresh perspective that people in an organization do not possess.
- Supporting Expert
  - an analyst draws on professional expertise concerning computer hardware and software and their uses in the business.
- Agent of change

5 What's a System/Software Development Life Cycle?

> The systems development life cycle (**SDLC**) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application. **SDLC** can apply to technical and non-technical systems

Consists of five consecutive phases:

- Planning Phase

- Analysis Phase
- Design Phase
- Implementation / coding phase
- Maintenance Phase

5.1 Planning Phase

- Answers why the information System should be built
- Determine how the project team will **go**
- Composed of:
  - Project initiation
  - Project management
- Includes:
  - Project initiation
  - ensuring feasibility
  - Project Plan & schedule
  - Obtain needed approvals

5.2 Analysis Phase

The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used.

- Analysis Phase:
  - Answers the following:
    - Who will use the system
    - What hte system will do
    - Where it will be used
    - When It will be used
  - The Project team in this phase:
    - Investigate any current system
    - identify improvement
    - develop a concept for the new system
  - Composed of
    - Analysis Strategy
      - to guide project efforts
      - Analyze the current system if it exists
    - Requirements gathering
      - Leads the development of a concept for a new system
      - The concept is used to build a set of analysis models
    - System Proposal
      - Presented to Project sponsor
      - Sponsors decide if the project will continue Describes the basic business requirements the new system should meet.
    - Includes
      - Understanding business needs

        - obtaining functional requirements
    - Delivery:
        - Analysis
        - High-level initial design

5.3 Design Phase

- Define the solution system and how it will operate in terms of:
  - Hardware
  - Software
  - Networks
- Define the:
  - User interface
  - Forms used
  - Reports used
- Identify:
  - Needed files
  - Needed Programs
  - Needed databases and data

5.4 Implementation/ Coding Phase:

- The system is developed or purchased
- Is considered the longest phase.
- The most expensive phase
- Composed of:
  - System construction
    - Construction of the system
    - Testing of the system
  - Installation
    - Support installed systems
    - post-implementation review
  - Maintenance Plan
- Includes
  - Training the users
  - Installation of the new system

5.5 Maintenance Phase

- Is done periodically
- Done On hardware and software
- done to debug errors
- Keep improving and upgrading:
  - e.g. add new functionality

6 Systems Development Life Cycle Models

6.1 Waterfall Model

- Traditional sequential pioneer **SDLC**
- Became the defining engineering approach to SW development
- done in phases and each phase has various tasks and objectives
- Deliver complete product at the end.
- Characterized by:
  - Feedback Loops
  - Documentation-driven
- **Advantages** (pros):
  - Provides structured approach
  - Sets requirements early
  - Easy
  - Milestone are better understood
- **Disadvantages** (cons):
  - The working version will be available late.
  - has blocking states
  - specification are
    - Long
    - detailed
    - Written in a style unfamiliar to the client

6.2 Rapid Prototyping Model

- offers a small-scale replica working "prototype" of the product and then obtains customer feedback to refine the prototype.
- delivers a complete product at the end
- sub-phases for the prototype:
  - Establish objective → plan
  - define functionality → outline definition
  - develop → product
  - evaluate → evaluation report.
- Characteristics:
  - used in requirements phase
  - evaluated by the customer
- **Disadvantages** (cons)
  - could be discard-do not turn into a product
  - not proven
- *Recommendation* : use the **rapid prototyping model (RAD)** in defining requirements and **waterfall** in the rest of the phases.

6.3 Incremental Model

- requirements are broken down into multiple independent modules

- each iteration passes through 4 phases.
- deliver a portion of the product at each stage
- takes from $5 \rightarrow 25$ iteration to build a product
- **Advantages** (pros)
  - **SW** is generated quickly
  - flexible
  - less costly
  - less expensive to change requirements and scope
  - customers can respond to each building module
  - errors are easy to be identified
- **Disadvantages** (cons)
  - requires good planning and designing
  - system architecture causes problems
  - the iteration phase is rigid
  - time-consuming in debugging
- *When to use*
  - clear requirements
  - need for early releases.
  - no enough skilled engineers
  - There are hight risk features and goals
  - e.g. web applications and product based companies

6.4 Spiral Model

- is risk-driven **SDLC** model
- is based on risk patterns that push the team to adopt one or more process models
- each phase is:
  - proceeded by
    - alternative
    - *risk analysis*: identification of potential risk is done while risk mitigation strategy is planned and finalized
  - followed by
    - evaluation
    - planning for the next phase
- steps:
  - objectives and alternative solutions determination
  - identify/resolve risks
  - develop next version
  - review and plan for the next phase
- *when to use*
  - for a project that is/has :
    - large
    - requires frequent releases
    - prototypes applicable
    - medium to hight-risk projects

        * unclear and complex requirements
    - When
        * risk and cost evaluation is important
        * changes may occur at any time
        * long-term project commitment is not feasible due to changes in economic priorities
- **Advantages** (pros)
    - give space for customer feedback
    - fast development
    - features are added systematically
    - repeated development helps in risk management
    - cost analysis is easier due to prototype fragmentation
    - additional functionality can be done at a later stage
- **Disadvantages** (cons)
    - risk of not meeting budget/schedule
    - demands risk assessments expertise
    - should be followed strictly
    - needs more documentation
    - not good for small projects

6.5 Agile process model

- combines:
    - philosophy
        * encourages
            · customer satisfaction
            · early incremental deliver of the software
        * small highly motivated team
        * informal methods
        * minimal software engineering work
        * development simplicity
        * people and interaction are emphasized rather than processes and tools
    - Development guidelines
        * stresses *deliver* over *analysis* and *design*
        * active/continuous communication between developers and customers
- Steps
    - Exploration
    - planning
    - first release (repeat) - involve customer before release
    - Producterizing (Repeat) - Increase the pace of iterations after the product is released
    - Maintenance then planning
- **Advantages** (pros)
    - High customer satisfaction

  - better at adapting to late changes in requirements
  - supports face-to-face conversation
  - better at adapting to changing circumstances
  - **SW** is delivered frequently i.e. in weeks
  - customers, developers and tester constantly interact with each other
  - could continue to move forward without fear of reaching a sudden standstill
  - gives more freedom of options and time for developers and stakeholders.
    * freedom of option given them the ability to leave important decisions until more data is available
- **Disadvantages** (cons)
  - only senior programmers can take decision in the **SDLC**
  - the project can easily get taken off track if the customer is not clear about what outcome they want
  - less emphasis on designing and documentation
  - in large project, difficult ot assess the errors
- *When to use*
  - When new changes need to be implemented

6.6 Extreme Programming (*XP*)

- description
  - variated from the incremental model
  - controversial new approach
  - based on pair programming
  - based on the continuous integration of tasks
- steps
  - project requirements
  - stories - features that client wants to estimate cost and time
  - test cases - client select stories each build (tasks)
  - tasks
  - completion
- features
  - computers are put in the center of a large room lined with cubicles
  - client representatives works with the **XP** team at all times.
  - There is no specialization
  - There is no overall design phase - refactoring
  - After completion, the **XP** team has iteration planning meetings with the customers
  - Tasks are accepted based on customer testing

6.7 How to select the suitable *SDLC* model?

- Each of these approaches is suitable for different projects, environment, and requirements

- e.g. **Waterfall** is good for simple projects. However, **Spiral** or **iterative** are better for large-scale projects
- **No single model is the best of all the *SDLC* models discussed**
- To select the best model
  - know all the types of **SDLC** models
  - Assess the requirements of all the stakeholders
  - Decide on a method that best fits your needs

6.7.1 Criteria for deciding on a model

- Product complexity
- Product size
- Magnitude of changes
- Frequency of changes
- Skills of the development team
- Time constraints
- Access to users

7 Requirements and Requirements engineering

7.1 Goal of Software development

- Develope quality software on time and on budge - that meets customers' real needs.
- customer needs are critical for software development
- Requirements shall be engineered and analyzed between testers, customers, designers, and implementers

7.2 Requirements

7.2.1 Definition

> Abstract description of the *services* provided by the system and the constraints under which the system must operate. Written *understandably* for the customers

7.2.2 Requirements Specification

- Produced in form of a requirement specification document
  - A structured document that sets out the system service in detail

7.2.3 Characteristics of good requirements

- Unambiguous
  - doesn't use short forms or abbreviations

  - doesn't not contain words or forms that might have a double meaning i.e. No need to think about a double meaning. i.e. don't make me think twice
- Verifiable
  - don't say "etc"
  - don't say "as they say, ..."
  - The arguments in the requirement should be exact.
- Concise
  - don't use run-on
  - don't use long *unnecessarily* long sentences.
  - don't use may conjunctions
- correct
  - no Falsified information
- understandable
  - Grammatically correct
  - don't use may,will,must or can
- Feasible/realistic
  - could be applied and tested
- independent
  - don't relate requirements to each other
- atomic
  - focusing on one point only or one element
- necessary
  - shall be needed by an stakeholder
  - shall be effective (can't go without it)
- Implementation-free (Abstract)
  - should not contain unnecessary design and implementation information
  - don't mention coding or files

7.2.4 Characteristics for the Set of Requirements

- Consistent
  - No conflicts between the requirements
  - applied terminology should consistent
- Non-redundant
  - No overlap between the requirement
- complete
  - all applicable requirements should be specified

7.2.5 Functional and non-functional requirements

7.2.5.1 Functional requirements

- Functional Requirements include:
  - Activity that the system must perform
  - Mention the process the system shall do

  - Mention the information the system shall include
  - flow directly into the creation of functional, structural and behavioral models.

7.2.5.2 Non-functional Requirements

- specifies
  - how the system should behave
  - system behavior constraints
- Quality attributes for a system

7.2.6 Difference between functional and non-functional requirements

The difference is that *non-functional requirement* describes **how the system works**, while *functional requirement* describes **what the system should do**

7.2.7 Requirements gathering techniques

- Observation
- Interview
- Questionnaires
- Scenario Walkthourghs
- Focus groups
- Prototypes (**RPD**)

8 Information Gathering

8.1 Interviews

The interview is the most commonly used requirements elicitation technique. After all, it is natural

- A one-on-one dig through your user's knowledge base
- A way to collect large amounts of in-depth data quickly
- Exploring someone's knowledge and needs in-depth, one-on-one
- Ensure that you understand the real, not just the perceived need

8.1.1 Planning for an interview

- Read background materials
- Establish objectives
- decide whom to interview
- Prepare the interviewee
- Decide Question types and Structures

8.1.2 Structured vs Non-structured interviews

8.1.2.1 Structured interviews

- Strictly adhere to the use of an interview protocol to guide
- Rigid interview style
- not a lot of opportunities to probe and further explore topics
- advantageous if there is a comprehensive list of interview questions

8.1.2.2 Semi-structured interviews

- use an interview protocol to help guide the interview
- maintain some structure
- ability to probe the participant for additional details
- offers flexibility
- no need for several rounds as it follows a protocol

8.1.2.3 Unstructured interviews

- happen with few if any, interview questions
- progress in the manner of a normal conversation on some topic
- formless interview style to establish rapport
- good to discuss sensitive topics
- need to conduct several rounds of interviews
- participant's narratives maneuver the conversation away

8.1.3 Recording the interview

8.1.3.1 Tape record

- need permission to be done
- **Advantages** (pros)
  - provide a completely accurate record
  - free the interviewer to listen and respond
  - allow better eye contact
  - allow better rapport
  - allow replay of the interview
- **Disadvantages** (cons)
  - make the interviewee nervous
  - less apt to respond freely
  - make the interviewer less apt to listen
  - difficulty to find important passages on a long tape
  - increasing costs of data gathering

8.1.3.2 Note taking

- **Advantages** (pros)
  - keeping the interviewer alert

  - aid recall of important questions
  - help recall important interview trends
  - show interviewer interest in the interview
  - demonstrate the interviewer's preparedness
- **Disadvantages** (cons)
  - loss vital eye contact
  - loss the train of conversation
  - make the interviewee hesitant to speak when notes are being made
  - causing excessive attention to fact and too little attention to feelings and opinions

8.1.4 Questions Types

8.1.4.1 Open-ended

- Allow interviewees to respond how thy with any length
- interested in the breadth and depth of the reply
- **Advantages** (pros)
  - put the interviewee at ease
  - allow the interviewer to pick up on the interviewee's vocabulary
  - reflect education, values, attitudes, and beliefs
  - richness of detail
  - Avenues of further questioning
  - more interesting for the interviewee
  - allows more spontaneity
  - Makes phrasing easier for the interviewer
  - useful if the interviewer is unprepared
- **Disadvantages** (cons)
  - result in too much irrelevant details
  - possibly losing control
  - take too much time for useful information
  - potentially seems that the interviewer is unprepared

8.1.4.2 Close-ended

- Limit the number of possible responses
- Generate precise, reliable data which is easy to analyze
- The methodology is efficient
- requires little skill for interviewers
- **Advantages**
  - Saving interview time
  - Easily comparing interviews
  - Getting to the point
  - keeping control of the interview
  - covering a large area quickly
  - Getting to relevant data
- **Disadvantages**

  - Boring for hte interviewee
  - Failure to obtain rich detail
  - Missing main ideas
  - Failing to build rapport between interviewer and interviewee

8.1.4.3 Bipolar Questions

- Answered with a ("yes","no") or ("agree","disagree")
- Should be used sparingly

8.1.4.4 Probing Questions

- Elicit more detail about *previous questions*
- Purpose
  - Get more meaning
  - Clarify
  - Draw out and expand on the interviewee's

8.1.5 Pitfalls

- Avoid leading questions as leading questions tend to guide interviewees into responses desired by the interviewer
  - should be avoided to
    - reduce bias
    - improve reliability
    - improve validity
- Avoid double-barreled questions as asking two questions at once
  - should be avoided
    - as interviewees may answer only one question
    - leading to difficulties in interpretation

8.1.6 Questions Structuring

8.1.6.1 Pyramid

- Begin with detailed questions (Close-ended Questions)
- Expand by allowing generalized responses (Open-ended questions)
- Useful if
  - Interviewees need to be armed up to the topic
  - seem reluctant to address the topic

8.1.6.2 Funnel

- Begin with generalized questions (Open-ended questions)
- Conclude by narrowing down (Close-ended questions)
- it provides an easy, non-threatening way to begin an interview
- good when the interviewee feels emotional about the topic

8.1.6.3 Diamond

- Begins in aver specific way (Close-ended questions)
- Then more general issues (open-ended questions)
- Conclude (close-ended questions)
- *useful in* keeping the interviewee's interest and attention through a variety of questions

8.1.7 Phases

8.1.7.1 Before

- contact the interviewee and confirm the interview
- dress appropriately
- arrive a little early
- affirm that you are present

8.1.7.2 During

- the interview should not exceed 45 minutes to one hour
- make sure that you are understanding what the interviewee is telling you
- ask for definitions if needed
- use probing questions

8.1.7.3 After

8.1.7.3.1 Closing

- Always ask "is there anything else that you would like to add?"
- summarize and provide feedback on your impressions
- ask whom you should talk with next
- set up any future appointments
- thank them for their time and shake the hand

8.1.7.3.2 Report

- write as soon as possible after the interview
- provide an initial summary, then more details
- review the report with the respondent

8.2 Joint Application Design (JAD)

an information gathering technique that allows the project team, users, and management to work together to identify requirements for the system.

- An information gathering technique that replaces interviews with the use community.

- Allows the analyst to accomplish requirements analysis and design the UI with users in the a group setting
- used when organizational culture supports joint problem-solving behaviors

8.2.1 Jad Personnel

- Analysts
- Users
- Executives
- Observers
- A scribe
- A session leader

8.2.2 Advantages

- A time saver compared to traditional interviews
- Rapid development of systems
- Improved user ownership of the system
- Creative idea production is improved

8.2.3 Disadvantages

- jad requires a large block of time
- If preparations are incomplete, the session may not go very well
- if the follow-up report is incomplete, the session may not be successful
- The organizational skills and culture may not be conducive to a JAD session

8.3 Survey/Questionnaire

- Useful in gathering information from key organization members about
  - attitudes
  - beliefs
  - behavior
  - characteristics
- Questions must not be uncertain
- Apply by
  - know what you are attempting to accomplish precisely with the study
  - offer to the correct individuals
  - determine due date

8.3.1 Questionnaire languages

- misunderstanding of inquiries can prompt useless and pointless answers by being
  - simple
  - specific

  - short
  - not patronizing
  - free of bias
  - avoiding objectionable questions

8.3.2 Scaling

- Use measurement scales
  - scaling is the process of assigning numbers to an attribute
  - reliability of scales refers to consistency in response
  - validity is the degree to which the questions measures what the analyst intends to measure

8.3.2.1 Nominal scales

Nominal scales are used to classify things in categories

8.3.2.2 Ordinal Scales

Allow classification & also imply rank ordering

8.3.2.3 Interval scales

- Used when intervals are equal
- There is no absolute zero
  - Meaning that the user has to classifying something, for example, as useful or not useful

8.3.2.4 Ratio scales

- The intervals between numbers are equal
- Have an absolute zero
  - Meaning that the user has to ability to select a ratio that is exactly zero.

8.3.3 Questionnaire format

8.3.3.1 Fixed Format

- Consists of questions that need a variety of predefined responses
- respondents have to choose an answer from a series of answers provided
- replies from this format of the questionnaire are a lot simpler to interpret
- increasingly latent; respondents can't ive their answers

8.3.3.2 Free format

- Enable users to answer openly for each inquiry
- respondent enters the appropriate response in the space

8.4 Observations

- observing
  - decision maker
  - decision maker's physical environment
  - interaction with ergonomic environment
    - all of these are important unobtrusive methods
- provides insight into what organizational members do
- confirms what has been found through other methods

8.4.1 Analyst play script

- involves observing the decision-maker's behavior and recording their actions using a series of action verbs as
  - talking
  - sampling
  - corresponding
  - deciding

9 DFD (Data flow diagram)

> *Graphical* representation that describes *processes* and *data movement* through out the system

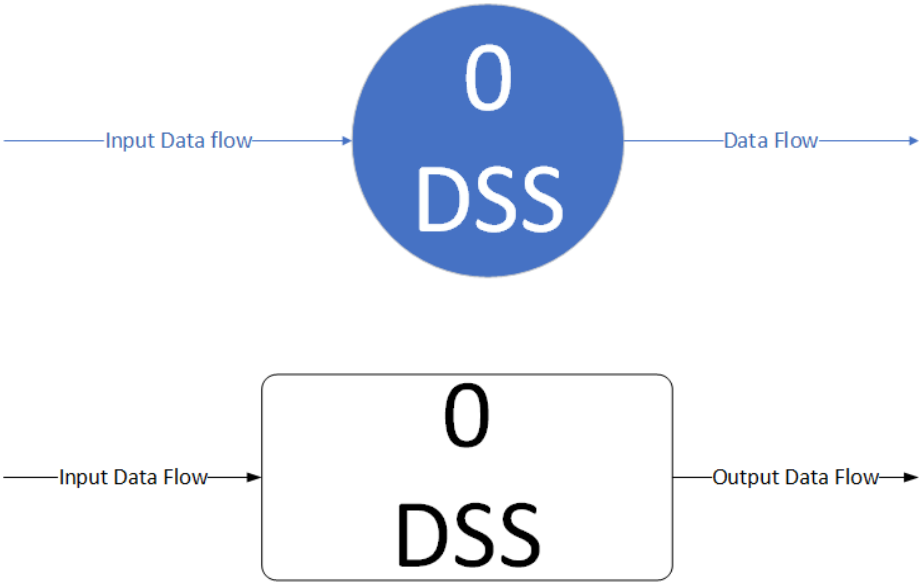
DFD consists of 4 elements:

- Process
- Data flow
- Data Store
- External entities

9.1 Process

Every **Process** must have

- A number
- A name
- at least one output data flow
- at least one input
- Can be exploded into *child* processes
- is represented by a rounded rectangles
  - can be also be represented by a circle



9.2 Data flow

Describes *movement* of *data* from one process to another Data is a single piece or a logical collection of several pieces of info.

Every *Data Flow* has:

- A name
- One or more connections
- must be in one direction.

9.3 Data Store

Collection of data stored in some way

Every *Data Store* has:

- A number (unique such as D_1, D_2, D_3)
- A name
- At least *one input* data flow
- at least *one output* data flows
- Shall be connect to processes

9.4 External entity

It's a **person,organization, or system** that is *external* to the *system*, but *interacts* with it (**provide** data or **receive** data)

Every *External entity* has:

- A name
- can't be connect to another entity
- Represented by a rectangle (two rectangle inside each other)

9.5 Context diagram

- To-level view of the system that shows the overall boundaries of it.
- Represents the results of fact-finding (information gathering)
- Contains one process numbered 0 which represents the whole system.
- Data flow connects the process to the external entities
- No data-stores
- No sub-processes

9.6 0-Diagram (Level 0 Diagram)

- Shows all the *Major processes* that build the system and how they are *interrelated*
- Process are numbered from $1, 2, \cdots$
- The major data stores and external entities are included
- Shows the data movement from and to each *process*
- There is one and only one level-0 Diagram

10 Data Dictionary

main method for analyzing the data flow and data stores of **DFD**
Data Dictionary is built for *each data flow* or *Data Store*.

Reason for using a data Dictionary

- Determine the contents of *data stores*
- Develop the **Logic** for data flow diagram processes
- Provide a starting point for developing screen and reports
- Validate the data flow diagram for **completeness** and **Accuracy**
- Eliminate redundancy

Data Dictionary consists of:

1. Description form
2. Data structures
3. Data Elements

10.1 Description form of Data flow

- Id
- unique descriptive name
- Description
- The source of the data flow
- The destination of the data flow
- Type of data flow
- The name of the data structure describing the elements
- The volume per unit time
- Comments

```
Id:                 1
Name:               Employee information
Description:        Contains full unchecked information about employees
Source              External Entity - Employee
Destination:        Process 1 - Create employee record
Type of Data flow:  Screen
Data Structure:     Employee Data
Volume/Time:        10/hour
Comments:           This information for one employee in company that request paycheck
```

10.2 Data Structure

- is a group of smaller **structures** and **elements**
- is represented by **Algebraic notation**
  - "=" → consists of
  - "+" → and
  - "{}" → group of elements (repetitive elements)
  - "[]" → either or elements
  - "()" → optional element

10.2.1 Examples

```
Employee Data =     Employee ID +
                    Employee Name +
                    Employee Address +
                    (Employee Email) +
                    Employee Department +
                    {Employee projects} +
                    [pay method]
```

10.2.2 Structural Records

- A structure my consist of elements or structural records
  - Employee Name
  - Employee Address

- Each of these must be further defined until they are broken down into their component elements

10.2.2.1 Example

```
Employee Name       =   First Name +
                        (middle) Name +
                        Last name
Employee Address    =   Street +
                        (Apartment) +
                        City +
                        State +
                        Zip +
                        (country)
```

10.3 Logical And physical data structures

10.3.1 Logical

Shows what data the business needs for its day-to-day operations.

10.3.2 Physical

Include additional elements necessary for implementing the system

- Key fields used to locate records
- codes used to identify record status
- Transaction codes to identify different record types
- Repeating group entries
- Limits on items in a repeating group
- Password.

10.4 Data Element

Should be defined with **Descriptive information**, *length* and *type of data information*, *validation* and *default values*

Elements should be defined once in the data dictionary

Data element consists of:

- Element Id
- Name
- Alias
- Short description
- Base or derived?
- Element length
- Type of data
- Input and output formats

- Validation criteria
- default value (if it exists)
- Comments