CS5002NP

Software Engineering

Agendas

- **Data Dictionary**
- **Entity Relationship Diagram**
- **Structure Chart**

Data Dictionary

- Collection of **names**, **definitions**, and **attributes** about data elements that are being used or captured in a database, information system, or part of a research project

- Describes the **meanings** and **purposes** of data elements within the context of a project, and provides guidance on interpretation, accepted meanings and representation

Data Dictionary

- Provides **metadata** about data elements
- Metadata can assist in **defining the scope** and **characteristics of data elements**, as well the rules for their usage and application.

Data Dictionary - Uses

- Assist in **avoiding data inconsistencies** across a project
- Help **define conventions** that are to be used across a project
- **Provide consistency** in the collection and use of data across multiple members of a research team
- Make data **easier to analyze**
- Enforce the use of **Data Standards**

Data Dictionary

- **Lists all data items** appearing in the DFD model of a system
- The data items listed include **all data flows and the contents** of all data stores appearing on the DFDs in the DFD model of a system
- A data dictionary lists the purpose of all data items and the definition of all composite data items in terms of their component data items

Data Dictionary

- For example,
  A data dictionary entry may represent that the data **grossPay** consists of the components **regularPay** and **overtimePay**

grossPay = regularPay + overtimePay

Data Dictionary

- For the **smallest units of data items**, the data dictionary lists **their name and their type**
- **Composite data items** can be defined in terms of **primitive data items** using different **data definition operators**

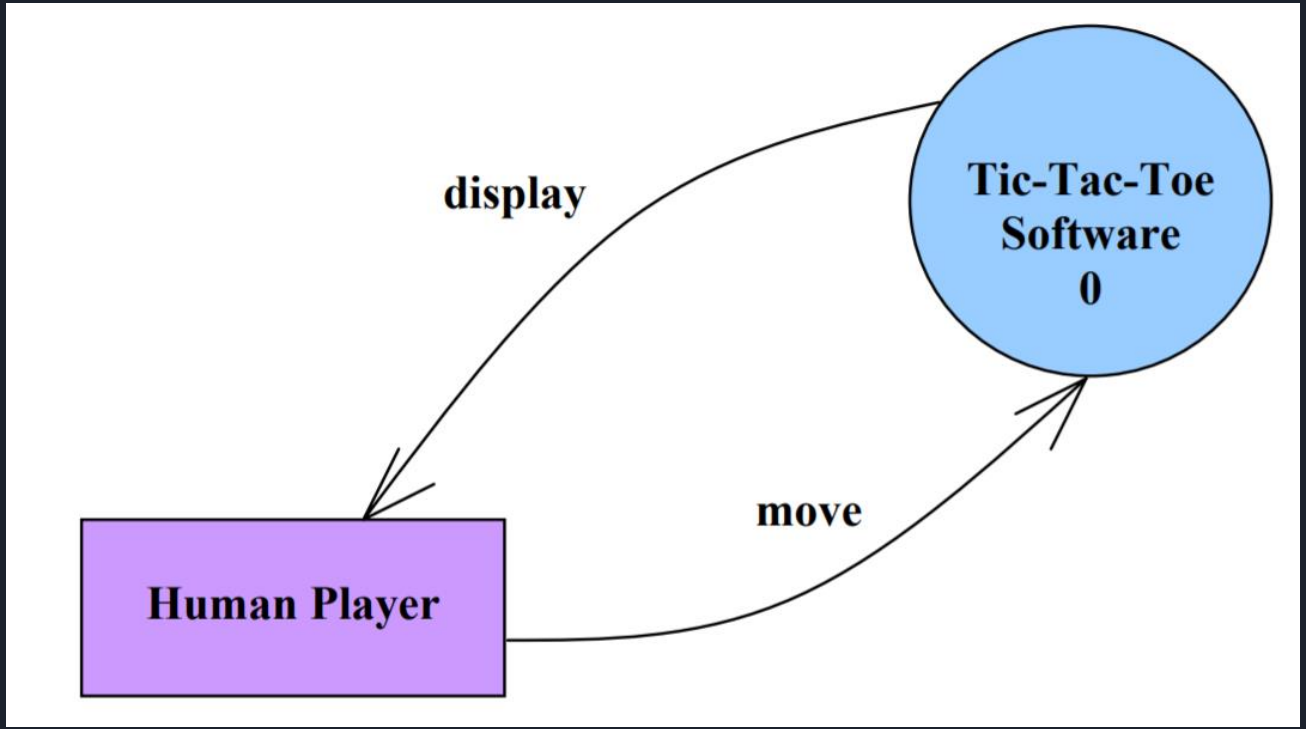
Data Dictionary - Operators

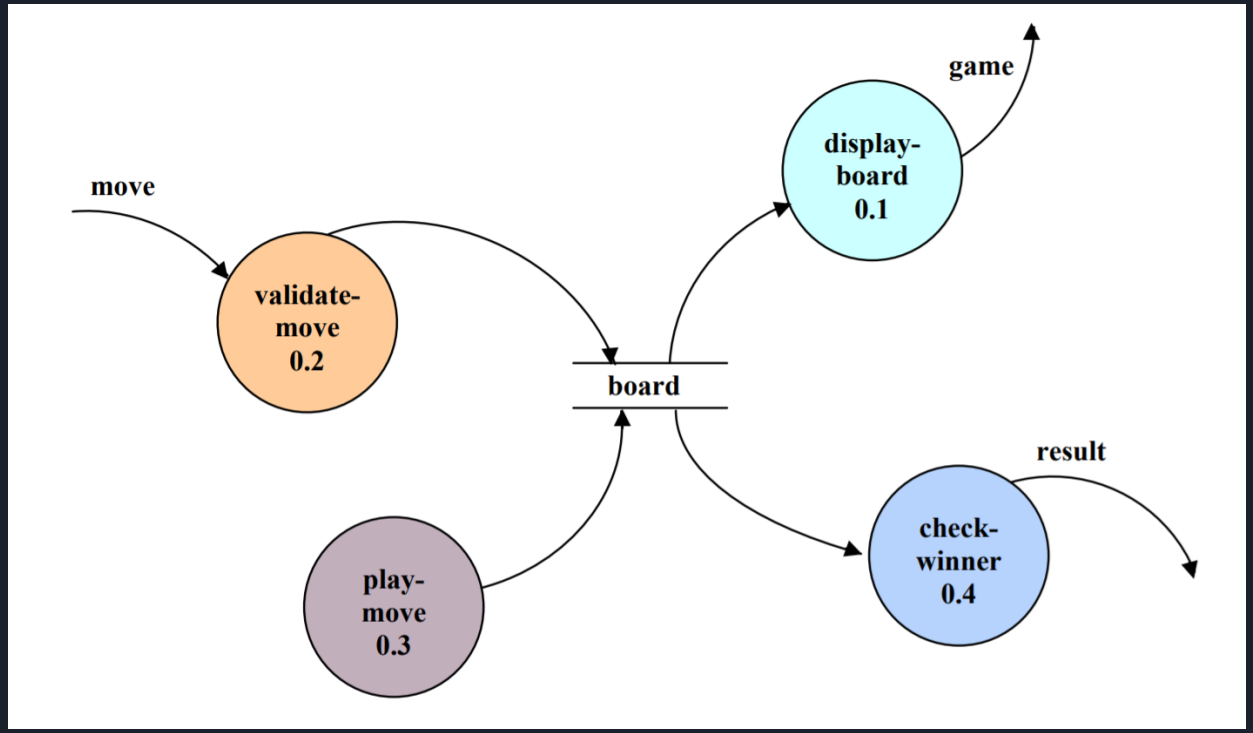
- **+** : denotes composition of two data items, e.g. a+b represents data a and b
- [, ,] : represents selection, i.e. any one of the data items listed in the brackets can occur. For example, [a,b] represents either a occurs or b occurs
- ( ) : the contents inside the bracket represent optional data which may or may not appear. e.g. a+(b) represents either a occurs or a+b occurs

Data Dictionary - Operators

- { } : represents iterative data definition, e.g. {name}5 represents five name data
- {name}* represents zero or more instances of name data
- = : represents equivalence, e.g. a=b+c means that a represents b and c
- /* */ : Anything appearing within /* and */ is considered as a comment.

Data Dictionary - Tic-Tac-Toe


display
Tic-Tac-Toe
Software
0
move
Human Player

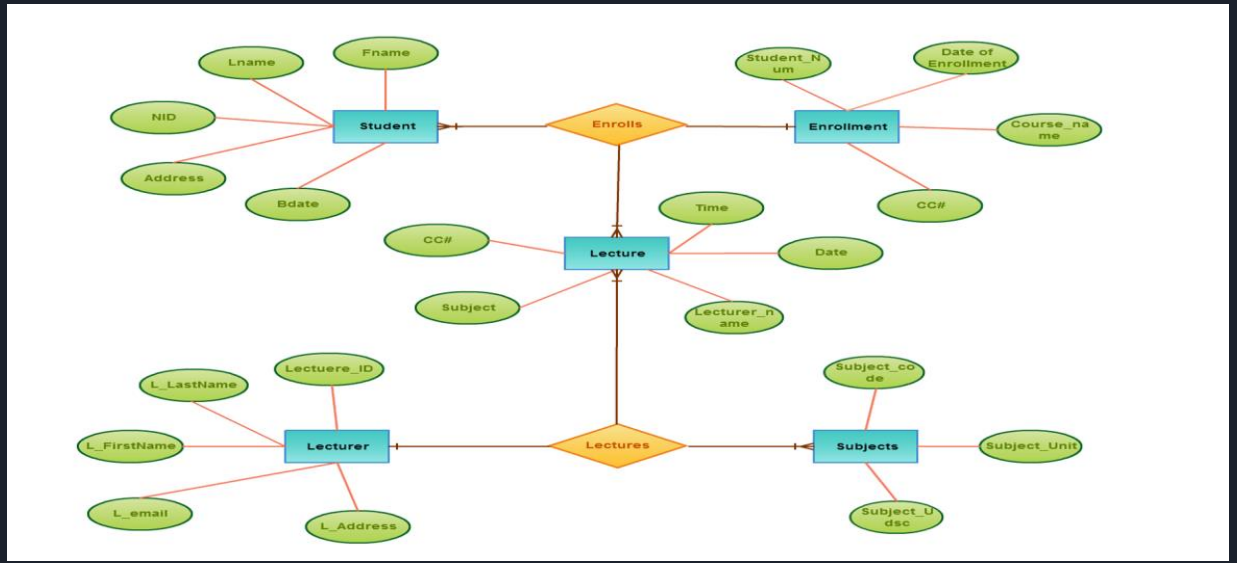

move
validate-
move
0.2
board
display-
board
0.1
game
play-
move
0.3
check-
winner
0.4
result

A single data dictionary should **capture all the data appearing** in all the DFDs constituting the model

Data Dictionary - Tic-Tac-Toe

- **move**: integer /*number between 1 and 9 */
- **display**: game+result
- **game**: board
- **board**: {integer}9
- **result**: [“computer won”, “human won”, “draw”]

Entity Relationship Diagram



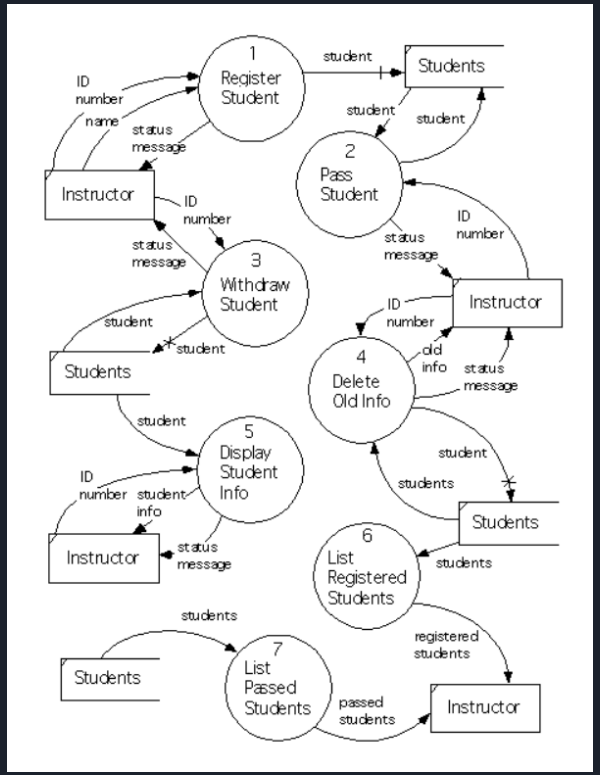
Process Specification - PSpecs

- A method used to **document**, **analyze** and **explain** the **decision-making logic** and **formulas** used to create output data from process input data
- **Objective** - flow down and specify regulatory/engineering requirements and procedures

Process Specification - PSpecs

- Reduces **ambiguity**, allowing an individual or organization to obtain a precise description of executed tasks and accomplishments and validate system design, including the data dictionary and data flow diagrams

- High-quality, consistent data requires **clear and complete process specifications**


1
Register Student
2
Pass Student
3
Withdraw Student
4
Delete Old Info
5
Display Student Info
6
List Registered Students
7
List Passed Students
Students
Instructor
ID number
name
status message
student
old info
students
student info
registered students
passed students

PSpecs - Format

Process Name: Register Student
Process Number: 1
Inputs: ID number, name
Outputs: status message, student
Process Logic:

- Staff Supplies students details
- Student data is stored in the data store
- If ID number is invalid, a failed status message will be given
- If ID number is valid, a success status message with student info will be given

PSpecs - Format

Process Specification Form

Number 1.5
Name Determine Quantity Available
Description Determine if an item is available for sale. If it is not available, create a backordered item record. Determine the quantity available.

Input Data Flow
Valid Item from Process 1.2
Quantity on Hand from Item Record

Output Data Flow
Available Item (Item Number + Quantity Sold) to Processes 1.4 & 1.5
Backordered Item to Inventory Control

Type of Process
☑ Online ☐ Batch ☐ Manual

Subprogram/Function Name

Process Logic:

```
IF the Order Item Quantity is greater than Quantity on Hand
    Then Move Order Item Quantity to Available Item Quantity
         Move Order Item Number to Available Item Number
ELSE
    Subtract Quantity on Hand from Order Item Quantity
        giving Quantity Backordered
    Move Quantity Backordered to Backordered Item Record
    Move Item Number to Backordered Item Record
    DO write Backordered Record
    Move Quantity on Hand to Available Item Quantity
    Move Order Item Number to Available Item Number
ENDIF
```

Refer to: Name:
☐ Structured English ☐ Decision Table ☐ Decision Tree

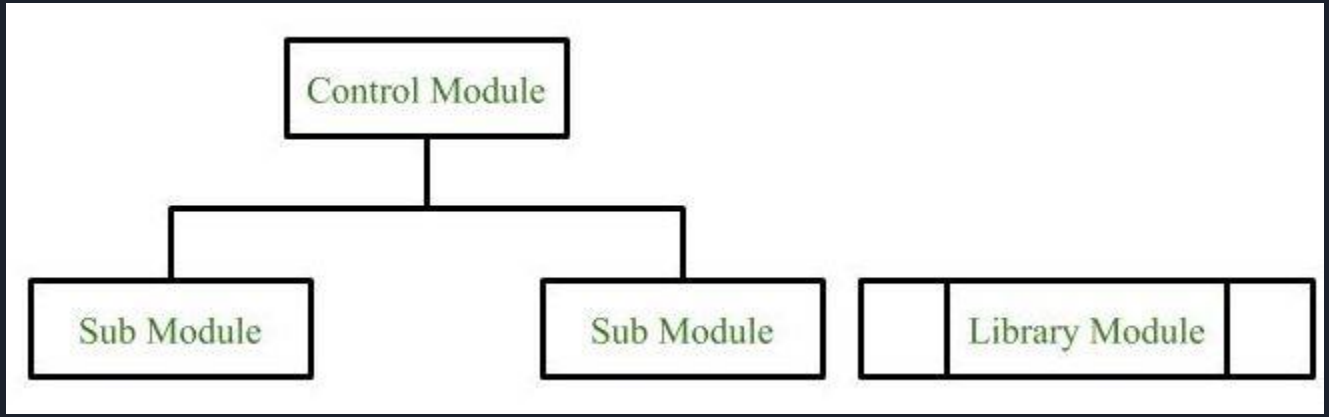
Structure Chart

- Represent **hierarchical structure** of modules
- **Breaks down** the entire system into **lowest functional modules**, describe **functions** and **sub-functions** of each module of a system to a greater detail
- Partitions the system into **black boxes** (functionality of the system is known to the users but inner details are unknown). **Inputs** are given to the **black boxes** and **appropriate outputs** are generated.

Structure Chart - Symbols

- **Module**
  Represents the process or task of the system
  - **Control Module:** A control module branches to more than one sub module
  - **Sub Module:** Sub Module is a module which is the part (Child) of another module
  - **Library Module:** Library Module are reusable and invokable from any module

Structure Chart - Symbols

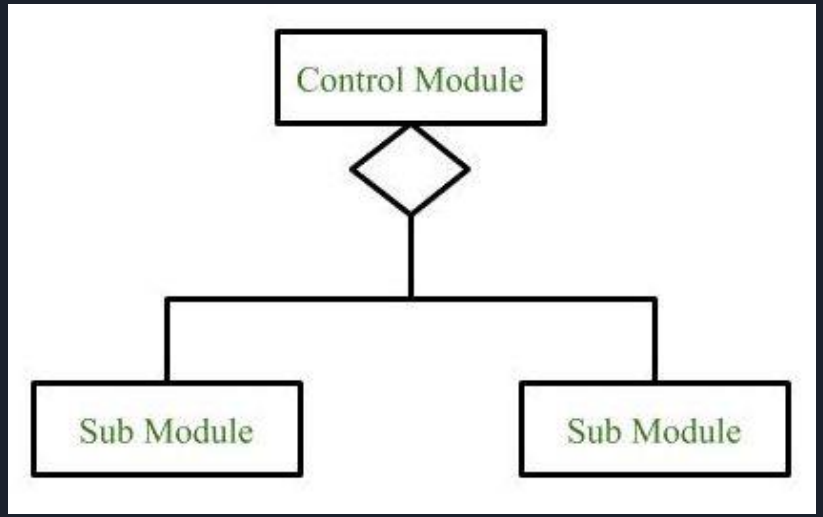


Structure Chart - Symbols

- **Conditional Call**

  Represents that control module can select any of the sub module on the basis of some condition

Structure Chart - Symbols



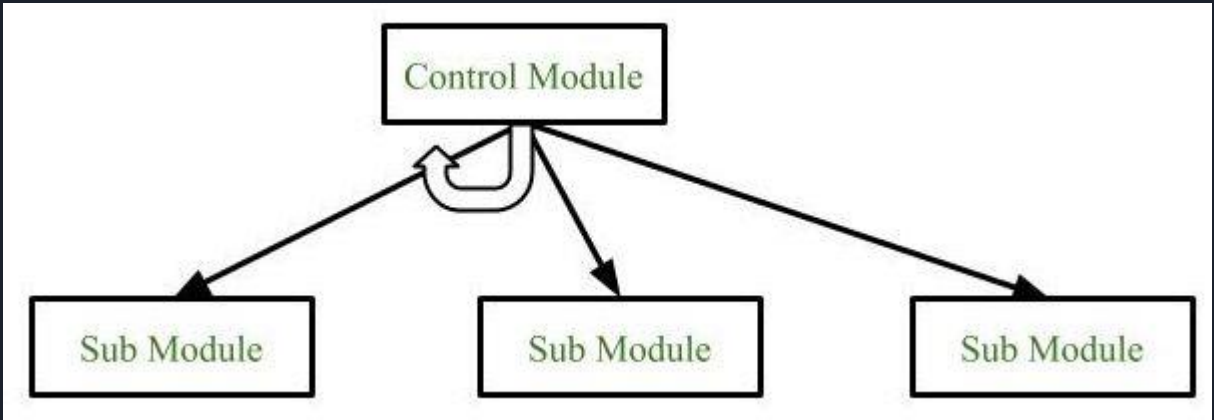
Structure Chart - Symbols

- **Loop**

  Represents the repetitive execution of module by the sub module

  A curved arrow represents loop in the module

Structure Chart - Symbols



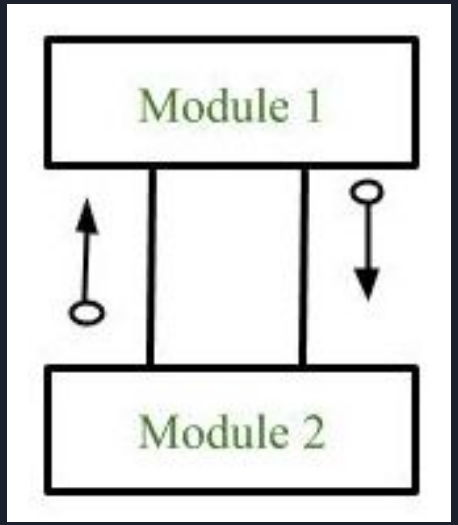
Structure Chart - Symbols

- **Data Flow**

  Represents the flow of data between the modules

  It is represented by directed arrow with empty circle at the end

Structure Chart - Symbols

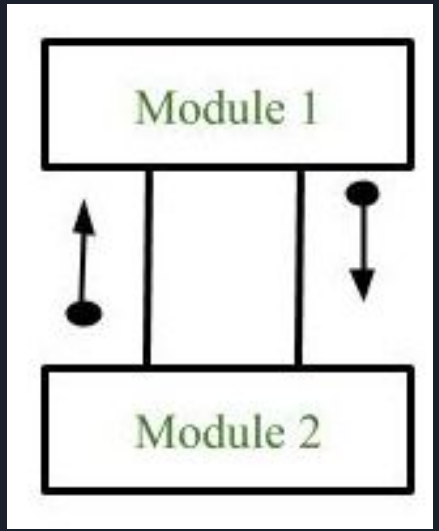


Structure Chart - Symbols

- **Control Flow**

  Represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end

Structure Chart - Symbols



Structure Chart - Symbols

- **Physical Storage**

  Where all the information are to be stored

THANK YOU!