Requirements Engineering

Semi-Formal Specification:
Structural Functional Requirements – Structured Analysis

- Data Flow Diagrams
- SADT
- IDEF0
Types of Requirements along different views

Functional Requirements (FRs)
- Structural Functional Requirements
  - Functional, i.e., Function-oriented
  - Informational, i.e., Information-oriented
- Behavioral Functional Requirements
- Non-Functional Requirements (NFRs)
Function Oriented Problem Analysis

- Creates a hierarchy of functions
  - Also called (process, activity, work-step, transactions, transforms, bubbles)
- Root is most abstract
- Leaf nodes the most detailed
- Most methods use data flow diagrams and dictionaries
- Examples
  - SRD structured requirements definition
  - SADT
  - Information Engineering
  - Modern structured Analysis
  - Problem statement Language
A data flow diagram (DFD) is a tool (and type of process model) that depicts the flow of data through a system and the work or processing performed by that system. DFDs have become a popular tool for business process redesign.
Data Flow Diagrams

- Existed long before computers
- Show the flow of data through a system

System
- Organization
- Company
- A computer hardware system
- A software system

Icons
- Data on the move – named arrows
- Transformations of data – named bubbles
- Sources and destinations of data – named rectangles (terminators)
- Data in static storage – two parallel lines
Data Flow Diagram Notations

**Process**
A process transforms incoming data flow into outgoing data flow.

**Yourdon and Coad Process Notations**

**Data Store**
Data stores are repositories of data in the system. They are sometimes also referred to as files.

**External Entity**
External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs.

**Gane and Sarson Datastore Notations**

**Dataflow**
Dataflows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
DataFlow Diagrams
Data Flow Diagram Layers

- Data flow diagrams are drawn in several nested layers.
- A single process node on a high level diagram can be expanded to show a more detailed data flow diagram.
- Draw the context diagram first, followed by various layers of data flow diagrams.
Context Diagrams

- A context diagram is a top level (also known as Level 0) data flow diagram. It only contains one process node (process 0) that generalizes the function of the entire system in relationship to external entities.
The first level DFD shows the main processes within the system. Each of these processes can be broken into further processes until you reach pseudocode.
Level 0 Data Flow Diagram
Explosion of Process 4

4.1 Obtain student preferences

4.2 Check eligibility

4.3 Check course availability

4.4 Inform student of unavailability

4.5 Enroll student in classes

4.6 Create wait lists

Students

Requirements

Student records

Accepted students

List of students

Eligible students

Preferred course lists

Refused classes

Schedule

New course assignment request

Process 5

Wait list offer

Individual registration information

Request for waiting list status
Data Flow Diagrams

- **Rules**
  - All names must be unique
  - Not a flow chart – no order implied
  - No logical decisions
  - Don’t get bogged down in detail

- **Leveling**
  - Preserve the number of inputs and outputs between the levels
Difference between DFD and Flowcharts

- Processes on DFDs can operate in parallel (at-the-same-time)
  - Processes on flowcharts execute one at a time

- DFDs show the flow of data through a system
  - Flowcharts show the flow of control (sequence and transfer of control)

- Processes on one DFD can have dramatically different timing
  - Processes on flowcharts are part of a single program with consistent timing
Homework -

- Draw a DFD for an ATM
Illegal Data Flows

Illegal data flows:

- B1 ➡️ B2
- B1 ➡️ DS1
- DS1 ➡️ B1
- DS1 ➡️ DS2

Corrected data flows:

- B1 ➡️ B1
- B1 ➡️ DS1
- DS1 ➡️ B1
- DS1 ➡️ DS2

a process is needed to exchange data flows between external agents
a process is needed to update (or use) a data store
a process is needed to present data from a data store
a process is needed to move data from one data store to another

Copyright © 2000 The McGraw-Hill Companies. All Rights reserved
## Structured English context

<table>
<thead>
<tr>
<th>Construct</th>
<th>Sample Template</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequence of steps</strong> — unconditionally perform a sequence of steps.</td>
<td>[ Step 1 ]&lt;br&gt;[ Step 2 ]&lt;br&gt;...&lt;br&gt;[ Step n ]</td>
</tr>
<tr>
<td><strong>Simple condition steps</strong> — if the specified condition is true, then perform the first set of steps. Otherwise, perform the second set of steps. Use this construct if the condition has only two possible values. <em>(Note: The second set of conditions is optional.)</em></td>
<td>If [ truth condition ]&lt;br&gt;  then&lt;br&gt;  [ sequence of steps or other conditional steps ]&lt;br&gt;else&lt;br&gt;  [ sequence of steps or other conditional steps ]&lt;br&gt;End If</td>
</tr>
<tr>
<td><strong>Complex condition steps</strong> — test the value of the condition and perform the appropriate set of steps. Use this construct if the condition has more than two values.</td>
<td>Do the following based on [ condition ]:&lt;br&gt;  Case 1: If [ condition ] = [value] then&lt;br&gt;  [ sequence of steps or other conditional steps ]&lt;br&gt;Case 2: If [ condition ] = [value] then&lt;br&gt;  [ sequence of steps or other conditional steps ]&lt;br&gt;...&lt;br&gt;Case n: If [ condition ] = [value] then&lt;br&gt;  [ sequence of steps or other conditional steps ]&lt;br&gt;End Case</td>
</tr>
</tbody>
</table>
Structured English Rules

Multiple conditions – test the value of multiple conditions to determine the correct set of steps.

Use a decision table instead of nested if-then-else Structured English constructs to simplify the presentation of complex logic that involves combinations of conditions.

A decision table is a tabular presentation of complex logic in which rows represent conditions and possible steps, and columns indicate which combinations of conditions result in specific steps.

Although it isn't a Structured English construct, a decision table can be named, and referenced within a Structured English procedure.
# Decision Table Example

<table>
<thead>
<tr>
<th>Condition Stub</th>
<th>Condition Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>If</td>
<td></td>
</tr>
<tr>
<td>Customer is bookstore</td>
<td>Y</td>
</tr>
<tr>
<td>Customer is librarian/individual</td>
<td>Y</td>
</tr>
<tr>
<td>Order size &gt; 6 copies</td>
<td>Y</td>
</tr>
<tr>
<td>Order size 50 copies or more</td>
<td>Y</td>
</tr>
<tr>
<td>Order size 20 to 49 copies</td>
<td>Y</td>
</tr>
<tr>
<td>Order size 6 – 19 copies</td>
<td>Y</td>
</tr>
<tr>
<td>Then</td>
<td>Allow a 25% discount</td>
</tr>
<tr>
<td>Allow a 15% discount</td>
<td>X</td>
</tr>
<tr>
<td>Allow 10% discount</td>
<td>X</td>
</tr>
<tr>
<td>Allow a 5% discount</td>
<td>X</td>
</tr>
<tr>
<td>Allow 0% discount</td>
<td>X</td>
</tr>
</tbody>
</table>
**Structured English constructs**

<table>
<thead>
<tr>
<th>Construct Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-to-many iteration</strong></td>
<td>Repeat the set of steps until the condition is false. Use this construct if the set of steps must be performed at least once, regardless of the condition's initial value.</td>
</tr>
</tbody>
</table>
| **Repeat the following until** | [truth condition]:
|                         | [ sequence of steps or conditional steps ]
|                         | End Repeat |
| **Zero-to-many iteration** | Repeat the set of steps until the condition is false. Use this construct if the set of steps are conditional based on the condition’s initial value. |
| **Do While**  | [truth condition]:
|               | [ sequence of steps or conditional steps ]
|               | End Do |
| **- OR -**     |                                                       |
| **For**        | [truth condition]:
|               | [ sequence of steps or conditional steps ]
|               | End For |
1. For each CUSTOMER NUMBER in the data store CUSTOMERS:
   a. For each LOAN in the data store LOANS that matches the above CUSTOMER NUMBER:
      1) Keep a running total of NUMBER OF LOANS for the CUSTOMER NUMBER.
      2) Keep a running total of the ORIGINAL LOAN PRINCIPAL for the CUSTOMER NUMBER.
      3) Keep a running total of CURRENT LOAN BALANCE for the CUSTOMER NUMBER.
      4) Keep a running total of AMOUNTS PAST DUE for the CUSTOMER NUMBER.
   b. If the TOTAL AMOUNTS PAST DUE for the CUSTOMER NUMBER is greater than $100.00 then:
      1) Write the CUSTOMER NUMBER and all their data attributes as described in the data flow LOANS AT RISK.
      Else
      1) Exclude the CUSTOMER NUMBER and data from the data flow LOANS AT RISK.

Structured English is a language and syntax, based on the relative strengths of structured programming and natural English, for specifying the underlying logic of elementary processes on DFDs.
Data Dictionaries

- Used to augment the Data Flow Diagrams
- Repository
- Layout
  - Name of the item
  - Alias
  - Description/Purpose
  - Related data items
  - Range of values
  - Data flows
  - Data structure definition/form
Data Structures

- Specific arrangements of data attributes that define a single instance of a data flow
  - A sequence that occur one after another
  - A selection of one or more attributes from a set
  - A repetition of one or more attributes
- Most common data structure notation is Boolean algebraic notation

<table>
<thead>
<tr>
<th></th>
<th>“Consists of” or “is composed of”</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Means and and designates sequence</td>
</tr>
<tr>
<td>[...]</td>
<td>Only one of the attributes may be present – <em>selection</em> - Attributes separated by commas Either/or</td>
</tr>
<tr>
<td>{...}</td>
<td>Attributes may occur many times – <em>repetition</em> - Attributes separated by commas</td>
</tr>
<tr>
<td>(...)</td>
<td>Attributes in side are optional no value for some of the data flows</td>
</tr>
</tbody>
</table>
## Example Data Structure

<table>
<thead>
<tr>
<th>DATA STRUCTURE</th>
<th>ENGLISH INTERPRETATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER=</td>
<td>An instance of ORDER consists of:</td>
</tr>
<tr>
<td></td>
<td>ORDER NUMBER and</td>
</tr>
<tr>
<td></td>
<td>ORDER DATE and</td>
</tr>
<tr>
<td></td>
<td>Either PERSONAL CUSTOMER NUMBER or CORPORATE ACCOUNT</td>
</tr>
<tr>
<td></td>
<td>and SHIPPING ADDRESS (which is equivalent to ADDRESS)</td>
</tr>
<tr>
<td></td>
<td>and optionally: BILLING ADDRESS (which is equivalent to ADDRESS)</td>
</tr>
<tr>
<td></td>
<td>and one or more instances of:</td>
</tr>
<tr>
<td></td>
<td>PRODUCT NUMBER and</td>
</tr>
<tr>
<td></td>
<td>PRODUCT DESCRIPTION</td>
</tr>
<tr>
<td></td>
<td>QUANTITY ORDERED and</td>
</tr>
<tr>
<td></td>
<td>PRODUCT PRICE and</td>
</tr>
<tr>
<td></td>
<td>PRODUCT PRICE SOURCE and</td>
</tr>
<tr>
<td></td>
<td>EXTENDED PRICE ] N+</td>
</tr>
<tr>
<td></td>
<td>SUM OF EXTENDED PRICES and</td>
</tr>
<tr>
<td></td>
<td>PREPAID AMOUNT and</td>
</tr>
<tr>
<td></td>
<td>(CREDIT CARD NUMBER+ EXPIRATION DATE) (QUOTE NUMBER)</td>
</tr>
<tr>
<td>ADDRESS=</td>
<td>An instance of ADDRESS consists of:</td>
</tr>
<tr>
<td></td>
<td>optionally: POST OFFICE BOX NUMBER and</td>
</tr>
<tr>
<td></td>
<td>STREET ADDRESS and</td>
</tr>
<tr>
<td></td>
<td>CITY and</td>
</tr>
<tr>
<td></td>
<td>Either STATE or MUNICIPALITY and optionally: COUNTRY and POSTAL CODE</td>
</tr>
</tbody>
</table>

| ORDER NUMBER + | ORDER DATE+ |
| [ PERSONAL CUSTOMER NUMBER, CORPORATE ACCOUNT NUMBER]+ |
| SHIPPING ADDRESS= ADDRESS+ |
| (BILLING ADDRESS= ADDRESS)+ |
| 1 (PRODUCT NUMBER+ |
| PRODUCT DESCRIPTION+ |
| QUANTITY ORDERED+ |
| PRODUCT PRICE+ |
| PRODUCT PRICE SOURCE+ |
| EXTENDED PRICE ) N+ |
| SUM OF EXTENDED PRICES+ |
| PREPAID AMOUNT+ |
| (CREDIT CARD NUMBER+ EXPIRATION DATE) (QUOTE NUMBER) |

| ADDRESS= | (POST OFFICE BOX NUMBER)+ |
| STREET ADDRESS+ |
| CITY+ |
| [STATE, MUNICIPALITY]+ |
| (COUNTRY)+ |
| POSTAL CODE |
A Simple Process Model
Traditional Approaches to Enterprise Modeling
SADT (Structured Analysis and Design Technique)

• Background
  • in use since the mid-seventies
  • inspiration for many commercial tools
  • (DFD?) trademark of Softech, Inc.

• View
  • "System" refers to any enterprise/organization, physical, manufacturing, and SW system

• Context Analysis should involve
  • technical assessment: feasibility of system architecture
    • Are the components and inter-relationships technically realizable?
  • operational assessment: system performance in a working environment
    • Can the system perform task X in less than a week of time?
  • economic assessment: costs & impacts of system implementation & use
    • Can the system be built with $20M, 1000 SE’s, in 2 yrs?
SADT (Structured Analysis and Design Technique)

Requirements definition
encompasses all aspects of system development prior to actual system design

Current or Foreseen Conditions

Management Objectives

Operational Concepts

Define system functions and allocate

Personal Functions

Software Functions

Hardware Functions

- loan evaluation
- warning of low inventory level
- destroy confidential info.
- loan request date + 7 days < now.date
SADT (Structured Analysis and Design Technique)

SADT Primitives
Diagrammatic!

Control

Input

Label

Output

Mechanism

Boxes "composed" into a diagram and interconnected through arrows
Each diagram is decomposed into up to "six" other diagrams
Informal documentation (as with DFDs)
Two types of diagrams

Actigrams
boxes: happenings - activities, operations, processing, events
box label: start with a verb

Datagrams
boxes: things - entities, objects, data, information, substances
box label: noun
SADT (Structured Analysis and Design Technique)

Semantics of Arrows

*In an actigram*

- Inputs are data that are consumed by the activity
- Outputs are produced by the activity
- Controls influence the execution of an activity but are not consumed
- Mechanism is a processor (machine, computer, person) which makes the activity happen
Example: Modelling Software Process for the development world

- Controls influence the execution of an activity but are not consumed
- Inputs are data that are consumed by the activity
- Outputs are produced by the activity
- Mechanism is a processor (machine, computer, person) which makes the activity happen

SADT (Structured Analysis and Design Technique)
SADT (Structured Analysis and Design Technique)

Example: Modelling Software Process for the development world

Design Requirements

Preliminary Design Data

Separate Data Structure From Algorithm

Perform Data Structure Design

Perform Algorithm Design

Recommended Detailed Design

Design Engineer
SADT (Structured Analysis and Design Technique)

Example: Modelling Software Process for the development world
SADT (Structured Analysis and Design Technique)

Semantics of Arrows

*In an actigram*

- Inputs are data that are consumed by the activity
- Outputs are produced by the activity
- Controls influence the execution of an activity but are not consumed
- Mechanism is a processor (machine, computer, person) which makes the activity happen

*In a datagram*

- Inputs are activities that produce the data
- Outputs consume the data
- Controls influence the internal state of the data
- Mechanism is a device for storage, representation, impl., etc.
A Simple Data Model
A Function Model is a Representation of the Activities and Relationships Between Activities in an Existing or Planned System.
IDEF0 (Integration Definition for Function Modeling)

Background

• released in Dec., 1993
• the "reference model" for system/enterprise function modeling
• also in use for software process modeling
• Federal Information Processing Standard maintained by Dept. of Commerce, NIST (National Institute of Standards and Technology) & Computer Systems Laboratory
• based on ICAM (Integrated Computer-Aided Manufacturing) from the US Air Force Wright Aeronautical Laboratories
• Information Modeling (IDEF1X) uses ERD + generalization/specialization
• closely resembles "actigrams" of SADT

Stringent Rules

E.g., Boxes shall be sufficient to insert box names
rectangular in shape with square corners
drawn with solid lines
Arrows that bend shall be curved using only 90 degree arcs
shall be drawn in solid line segments
vertically or horizontally, not diagonally
Traditional Approaches to Enterprise Modelling

IDEF0 (Integration Definition for Function Modelling)

C Diagram syntax rules

E.g.,

- *control feedbacks shall be shown as "up and over"*

```
  1  2
```

- *input feedbacks shall be shown as "down and under"*

```
  1  2
```

- *mechanism feedbacks shall be shown as "down and under"*

```
  1  2
```
Terminology of IDEFØ

- Functions and activities
- Diagrams, Boxes, and Arrows
- ICOMs: Inputs, Controls, Outputs, and Mechanisms
- Arrows, links, relationships, and concepts
- Splits, Joins, Unbundling, Bundling, and Branching
- Decompositions
- Viewpoint, Purpose, and Context
- NIST (FIPS ) standard
What is IDEFØ?

- An IDEF method for modeling functions
  - Graphics (diagrams)
  - Text (glossary & narrative)

- Provides both a process and a language for constructing a model of the decisions, actions, and activities in an organization
What is an IDEFØ Model?

- A definition of activities and information
  - Within a particular Context
  - Having a consistent Viewpoint
  - For a particular Purpose

- Series of diagrams (that decompose a subject into manageable chunks)

- A foundation for requirements specification, design, and programming

- A useful record throughout the life-cycle of an enterprise
Example IDEFØ Diagram

- Customer Expectations
  - Needs
  - Alternative Technologies
  - Knowledge of Previous Design
  - Raw Material

- Establish Requirements (A1)
  - Understanding of Customer Requirements

- Design System (A2)
  - Contract for Tradeoff Decisions
  - Design

- Build System (A3)
  - Analysis Methods
  - Design Methods
  - Fabrication Methods

- Output: Product
Diagram Construction (1)

- Boxes represent functions

- Arrows represent real objects or data

![Diagram]

- CONTROL
- INPUT
- FUNCTION
- OUTPUT
- MECHANISM
Labels are words that name functions and data/real objects.

Function labels are verbs or verb phrases and are put in the center of the function box.

Data labels are nouns or noun phrases.

Data labels name the input, control, output, and mechanism arrows.
IDEFØ Function

- An Activity, Action, Process, or Operation
- A Description of “What Happens” in a Particular Environment
- Accomplished by People, Machines, Computers
- Labeled with an Active Verb or Verb Phrase
IDEFØ Functions (Activities)

Represented as a box in an IDEF0 Model.

First diagram has one Function which bounds the context of the Model. (A - 0 diagram)

Diagram has a maximum of 6 functions & a minimum of 3
IDEFØ Relationships (Between Functions)

- Represented as arrows
- AKA concepts
- Real objects, data, people, machines, and computers
ICOMs

- Inputs
- Controls
- Outputs
- Mechanisms
Inputs

- Real Objects or Data Needed to Perform a Function
- Objects or Data Transformed by a Function
- Labeled with a Noun or Noun Phrase

INPUTS  
FUNCTION
Objects or Data Produced as a Result of the Function

Labeled with a Noun or Noun Phrase

Input

Output
Control

- That which Governs the Accomplishment of the Function
- Things that Influence or Determine the Outputs
- Labeled with a Noun or Noun Phrase
Mechanism

- Person, Device, or Data which Carries out the Function
- The Means by which the Function is Performed
- Labeled with a Noun or Noun Phrase

![Diagram of Mechanism]

INPUTS  ->  FUNCTION  <->  CONTROLS  <---  MECHANISMS  <---  OUTPUTS
Box and Arrow Relations in a Diagram

(input)

(output)

(feed back output)

(to control)

(output to inputs)

(output to control)

(output to control)

(output to mechanism)

(output)

(input)

(arrows branching)

(split)
Arrows: "Branching"

Output can branch and be used by two functions simultaneously or sequentially.

Without labels we cannot tell how the branching occurs.

Once this data is supplied, functions 2 & 3 can operate simultaneously or sequentially.
Arrows: "Joining"

PROCURED ITEMS → PRODUCTION ITEMS → CONTROL PRODUCTION ITEMS & TOOLS → FINISHED SUB-PARTS
Arrows: "Feedback"
Bundling and Unbundling

Bundle: Concepts B and C are bundled to form concept A.

Unbundle: Concept A is unbundled into concepts B and C.
Bundles and Unbundles

Unbundle

- Management Directives
- Orders
- Deliver Products
- Transaction Entries
- Account Entries
- Perform Billing
- Invoices

Bundle

- Keep Records
- Files
- Customer Records
- Prices & Tax Tables
- Billing Entries

Files = Customer Records + Price & Tax Tables
Account Entries = Transaction Entries + Billing Entries
Bundles and Un-bundles: PCB ASSEMBLY

Management Directives

Process plan

Load board onto m/c A1

Bare boards

Apply solder paste A2

Paste applied board

Solder paste method

Placement method

Soldering completed data

Placement completed data

Assembly Records

Chip positioned board

Place chip on board A3

Process Plan = loading details + solder paste details + chip placement method

Assembly Records = soldering completed data + placement completed data
“Parent” Activities Represent a Higher Level of Abstraction than that of Their “Children”
Further Decomposition

Parent Diagram

Parent Activity

Child Diagram

A0

A1

A2

A3

A31

A32

A33

A34

A4

A41
Decomposition

- Establishes model hierarchy

- Functions are comprised of other functions

- Decompositions is a process of breaking down of the functions (level-by-level)

- Data consistency is required throughout the level-by-level decomposition breakdown
Complexity Simplification Technique Tunneled Arrows

Tunneled Arrows at Connected Ends
(Concept Does Not Appear on the Next Lower Level.)

Tunneled Arrows at Unconnected Ends
(Concept Does Not Appear on the Next Higher Level.)
Tunneling Example

This control will not appear on child diagram.

This control will still be designated as C3 on child diagram.

This output will not be shown on parent diagram.
Steps in Building a Model

1. Define Viewpoint, Purpose, and Context

2. Develop the Context Diagram (Putting the situation in context)

3. Decompose activities to fit scope of modeling task (complete modeling per rules, etc)

4. Develop glossary
Model Orientation!!!!

- **Context (Subject)**
  The Boundaries of the Subject Matter

- **Viewpoint (Bias)**
  The Perspective from which a Subject is Analyzed

- **Purpose (Objective)**
  The Reason(s) a Model is Created
Example - Context Diagram

Acquire Materials

Inventory Policy

Purchase policy

Stock Levels

Payments

Rejected Materials

Vendor

ABC Co.

A-0 Diagram
Example - Decomposition of the Context Diagram

A0 Diagram
Function Model for Planning and Implementing a Feature Extraction module

- **Purpose:** To obtain a better understanding of the various tasks involved in planning and implementation of a feature extraction module.

- **Context:** We will assume CAD model formats, process planning requirements and resources available (people and computers) are known. The FE module will be built using available existing resources (no new tools or software will be purchased).

- **Viewpoint:** that of an industrial / mfg engineer who has a background in designing / building software systems.