Requirements Analysis

- Requirements analysis
  - specifies software’s operational characteristics
  - indicates software's interface with other system elements
  - establishes constraints that software must meet

- Requirements analysis allows the software engineer (called an analyst or modeler in this role) to:
  - elaborate on basic requirements established during earlier requirement engineering tasks
  - build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed.
A Bridge

The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high.

Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system.

Delay consideration of infrastructure and other non-functional models until design.

Minimize coupling throughout the system.

Be certain that the analysis model provides value to all stakeholders.

Keep the model as simple as it can be.
Domain Analysis

- Define the domain to be investigated.
- Collect a representative sample of applications in the domain.
- Analyze each application in the sample.
- Develop an analysis model for the objects.

Data Modeling

- Examines data objects independently of processing
- Focuses attention on the data domain
- Creates a model at the customer’s level of abstraction
- Indicates how data objects relate to one another
What is a Data Object?

**Object** — something that is described by a set of attributes (data items) and that will be manipulated within the software (system)

- each instance of an object (e.g., a book) can be identified uniquely (e.g., ISBN #)
- each plays a necessary role in the system i.e., the system could not function without access to instances of the object
- each is described by attributes that are themselves data items

Typical Objects

- **external entities** (printer, user, sensor)
- **things** (e.g., reports, displays, signals)
- **occurrences or events** (e.g., interrupt, alarm)
- **roles** (e.g., manager, engineer, salesperson)
- **organizational units** (e.g., division, team)
- **places** (e.g., manufacturing floor)
- **structures** (e.g., employee record)
Data Objects and Attributes

A data object contains a set of attributes that act as an aspect, quality, characteristic, or descriptor of the object.

object: automobile
attributes:
- make
- model
- body type
- price
- options code

What is a Relationship?

relationship — indicates “connectedness”; a "fact" that must be "remembered" by the system and cannot or is not computed or derived mechanically

- several instances of a relationship can exist
- objects can be related in many different ways
ERD Notation

**One common form:**

```
object1 -------(0, m)------- relationship -------(1, 1)------- object2
```

**Another common form:**

```
object1 ------- relationship ------- object2
```

Object-Oriented Concepts

- Must be understood to apply class-based elements of the analysis model
- Key concepts:
  - Classes and objects
  - Attributes and operations
  - Encapsulation and instantiation
  - Inheritance
**Flow-Oriented Modeling**

Represents how data objects are transformed as they move through the system.

A data flow diagram (DFD) is the diagrammatic form that is used.

Considered by many to be an ‘old school’ approach, flow-oriented modeling continues to provide a view of the system that is unique— it should be used to supplement other analysis model elements.

---

**The Flow Model**

Every computer-based system is an information transform ....
**Analysis Model Structure**

Data model
- data objects
- relationships
- ERDs

Functional model
- data transforms
- DFDs

Behavioral model
- events and states
- STDs

----

**The elements: data flow**

```
+-------------------+
| dividend          |
+-------------------+
| a data flow       |
+-------------------+
  
| number            |
+-------------------+

values can be copied

| address           |
+-------------------+
| street address    |
+-------------------+
| city              |
+-------------------+
| state             |
+-------------------+
| post code         |
```

values can be split
**The elements: actors**

- **Actor**: an actor is a producer or a consumer of data flows.
- **Window**: an actor forms a source or a sink — and is sometimes termed a terminator.

**Actors are objects**

**The elements: Data Stores**

- **Data Store**: a data store is a place where data values can be stored and retrieved later.

**Data stores are objects**
The Functional Model: The DFD

A terminator is a producer or a consumer of data flows.

A data store is a place where data values can be stored and retrieved later.
**Nested Data Flow Diagrams**

- Patient
- Nurse
- Patient monitoring system

- Nurse
- Patient
- Log data
- Request for report
- Vital signs

**Patient Monitoring System**

- Patient
- Nurse
- Local Monitoring

- Central Monitoring

- Report generator

- Patient bounds
  - Vital signs bounds
  - Formatted patient data

- Update log

- Log data
  - Patient log

- Warning message
  - Report

- Request for report
  - Log data
Getting Started

Example

- Manufacturing cell software *controls* a *robot* by *generation* of position coordinates that are *transmitted* to the *robot*. An *operator* inputs commands that cause the manufacturing cell software to *read* positioning and control commands from an *NC command file*. Components to be assembled are held in *parts fixtures* that *activate robot* control functions once each *fixture* contains a part ...

Use *nouns* to isolate external entities, data items and stores
Use *verbs* to help isolate processes (bubbles)

Creating a Context Diagram

Level 0 Flow Model (also called a “context diagram”)
DFD Questions

Q: How does the NC unit software transform input to output?
A: Lower DFD levels will provide details.

Q: What are "operator commands”
A: Another notional tool - the data dictionary - will help to describe.

Q: Where are processing details?
A: A PSPEC ("structured English") and other notational tools are used.

Refining to Level 1
Process & Control Models

Data Conditions

The PSPEC can be:
- narrative
- PDL
- equations
- tables
- diagrams and/or charts

```
if absolute tank pressure > max pressure
then
    set above pressure to "true"
else
    set above pressure to "false";
begin conversion algorithm x-Ol a;
compute converted pressure;
end
endif
```
The Control Model

- the control flow diagram is "superimposed" on the DFD and shows events that control the processes noted in the DFD
- control flows - events and control items - are noted by dashed arrows
- a vertical bar implies an input to or output from a control spec (CSPEC) - a separate specification that describes how control is handled
- a dashed arrow entering a vertical bar is an input to the CSPEC
- a dashed arrow leaving a process implies a data condition
- a dashed arrow entering a process implies a control input read directly by the process
- control flows do not physically activate/deactivate the processes - this is done via the CSPEC

Control Flow Diagram

The CSPEC can be:
- state transition diagram
- state transition table
- decision tables
- activation tables

combinatorial spec

input to the CSPEC
output from the CSPEC

The CSPEC can be:
- state transition diagram
- state transition table
- decision tables
- activation tables

combinatorial spec
**Decision Tables: example**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed rate account</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Variable rate account</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Consumption &lt; 100 Kwh</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>Consumption &gt; 150 Kwh</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>Minimum monthly charge</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schedule A Billing</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schedule B Billing</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**The Data Dictionary**

- a quasi-formal grammar for describing the content of data that the software will process and create
- a notation for describing control data and the values that control data can take, e.g., “on”, “off”
- a repository that also contains “where-used” / “how used” information
- a notation that can be represented manually, but is best developed using CASE tools
Data Dictionary

NAME = TITLE + {FIRST_NAME} + LAST_NAME
TITLE = [Mr. | Ms.]
FIRST_NAME = {LEGAL-CHARACTER}
FIRST_NAME = {LEGAL-CHARACTER}
LEGAL-CHARACTER = [A-Z|a-z|'|-| ]

= is composed of
+ and
() optional
{} iteration
[ | ] either - or
* ..text ...* comment
{ }n n repetitions of
( ... ) delimits a comment

Data Dictionary Example

Build the requirements dictionary:

<table>
<thead>
<tr>
<th>Name:</th>
<th>telephone number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases:</td>
<td>phone number, number</td>
</tr>
<tr>
<td>Where/How used:</td>
<td>read-phone-number (input)</td>
</tr>
<tr>
<td></td>
<td>display-phone-number (output)</td>
</tr>
<tr>
<td></td>
<td>analyze-long-distance-calls (input)</td>
</tr>
<tr>
<td>Description:</td>
<td>telephone no. = [ local extension</td>
</tr>
<tr>
<td></td>
<td>outside no. = 9 + { service code</td>
</tr>
<tr>
<td></td>
<td>service code = [ 211</td>
</tr>
<tr>
<td></td>
<td>domestic no. = ( (0) + area code ) + local number</td>
</tr>
<tr>
<td></td>
<td>area code = <em>three numeral designator</em></td>
</tr>
<tr>
<td>Format:</td>
<td>alphanumeric data</td>
</tr>
</tbody>
</table>
Modern Structured Analysis

An Illustration

Structured Analysis

The environmental model
• Statement of purpose
• Context diagram
• Event list

The behavioral model
• Data flow diagrams
• Process Specification
Journal administration:
The computer based system is used for administration of the information necessary to publish the Scandinavian Journal of Information Systems (SJIS). This includes registration of new subscribers, billing, mailing, and registration of subscriber data.

The Environmental Model

Statement of Purpose

- A brief, concise textual statement of the purpose of the system.
- Clarifies the boundaries of the system - what will be taken care of by the system, and what will not?

The Environmental Model

Context Diagram
**The Environmental Model**
**Event List**

SJIS:

1. Person or institution enters subscription.
2. Agent enters subscription on behalf of a person or institution.
3. The bank reports a money transfer.
4. Bookkeeping receives details on payments.
5. Agent cancels subscription.
6. Issue is sent to subscriber.
7. Invoice is sent to agent.
8. Invoice is sent to subscriber directly.
9. Subscription is cancelled.
10. Subscriber pays amount due.
11. ..... 

---

**The Behavioral Model: Dataflow**
**Models of Event Responses**

A flow tells the system that an event has occurred.

- Process "name"
- X: Necessary data in order to process data flow X
- Y: Output to terminate
- Z: Stores data related to asynchronous and interdependent events. Used for communication between processes.
- p: Store
SJIS: Event 10
Subscriber pays amount due

Event Responses Are Combined Into One Data flow Diagram
Upward Leveling of Data Flow Diagrams

Processes 4, 5, and 6 are united into a new process 7.
Processes 4, 5, and 6 are pushed one level down.

Data Flow Hierarchy
Process Specification

FIND Subscriber in Subscribers based on Subscriber#. IF not found: ERROR(Subscriber unknown) ELSE
  FIND invoicedata for Subscriber#. IF no pending invoice: ERROR(No pending invoice) ELSE mark invoice as paid.
  WRITE debitinfo to bookkeeping.

At the lowest level a process is described in structured English, flowcharts or similar.

Tool Requirements

- Support in modeling complex and systems related problems.
- Support building correct, complete and consistent models.
- Support building a simple and understandable model of a system.
- Help balance between details and overview.
DFDs: A Look Ahead

Maps into

analysis model

design model

Maps into
Class-Based Modeling

- Identify analysis classes by examining the problem statement
- Use a “grammatical parse” to isolate potential classes
- Identify the attributes of each class
- Identify operations that manipulate the attributes

Analysis Classes

- **External entities** (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.
- **Things** (e.g., reports, displays, letters, signals) that are part of the information domain for the problem.
- **Occurrences or events** (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.
- **Roles** (e.g., manager, engineer, salesperson) played by people who interact with the system.
- **Organizational units** (e.g., division, group, team) that are relevant to an application.
- **Places** (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.
- **Structures** (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects.
Selecting Classes — Criteria

- retained information
- needed services
- multiple attributes
- common attributes
- common operations
- essential requirements

Class Diagram

Class name: System

Attributes:
- systemID
- verificationPhoneNumber
- systemStatus
- delayTime
- telephoneNumber
- masterPassword
- temporaryPassword
- numberTries
- program()
- display()
- reset()
- query()
- modify()
- call()
CRC Modeling

- Analysis classes have “responsibilities”
  - Responsibilities are the attributes and operations encapsulated by the class

- Analysis classes collaborate with one another
  - Collaborators are those classes that are required to provide a class with the information needed to complete a responsibility.
  - In general, a collaboration implies either a request for information or a request for some action.
CRC Modeling

<table>
<thead>
<tr>
<th>Class: FloorPlan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>Responsibility:</td>
</tr>
<tr>
<td>Collaborator:</td>
</tr>
</tbody>
</table>

- Defines floor plan name/type
- Manages floor plan positioning
- Scales floor plan for display
- Incorporates walls, doors and windows
- Shows position of video cameras

Wall
Camera

Specification Guidelines

- Use a layered format that provides increasing detail as the "layers" deepen
- Use consistent graphical notation and apply textual terms consistently (stay away from aliases)
- Be sure to define all acronyms
- Be sure to include a table of contents; ideally, include an index and/or a glossary
- Write in a simple, unambiguous style (see "editing suggestions" on the following pages)
- Always put yourself in the reader's position, "Would I be able to understand this if I wasn't intimately familiar with the system?"
Specification Guidelines

Be on the lookout for persuasive connectors, ask why?
keys: certainly, therefore, clearly, obviously, it follows that ...

Watch out for vague terms
keys: some, sometimes, often, usually, ordinarily, most, mostly ...

When lists are given, but not completed, be sure all items are understood
keys: etc., and so forth, and so on, such as

Be sure stated ranges don't contain unstated assumptions
e.g., Valid codes range from 10 to 100. Integer? Real? Hex?

Beware of vague verbs such as handled, rejected, processed, ...

Beware "passive voice" statements
e.g., The parameters are initialized. By what?

Beware "dangling" pronouns
e.g., The I/O module communicated with the data validation module and its control flag is set. Whose control flag?

When a term is explicitly defined in one place, try substituting the definition for other occurrences of the term

When a structure is described in words, draw a picture

When a structure is described with a picture, try to redraw the picture to emphasize different elements of the structure

When symbolic equations are used, try expressing their meaning in words

When a calculation is specified, work at least two examples

Look for statements that imply certainty, then ask for proof keys; always, every, all, none, never

Search behind certainty statements—be sure restrictions or limitations are realistic
The Planning Game

- Business writes a story describing desired functionality
- Stories are written on index cards
- Development estimates stories
- Velocity determines number of stories per iteration
- Business splits and prioritizes stories and determines the composition of releases
- Velocity is measured and adjusted every iteration
- Customer steers development

Planning Game

User stories = lightweight use cases
2-3 sentences on a file card that:
- The customer cares about
- Can be reasonably tested
- Can be estimated & prioritized
Stories are promises for Conversation

• Stories are made up of two components
  • The written card
  • The series of conversations
    • Between customer and programmers
• The conversation will be captured as additional documentation that will be attached to the story
  • Design sessions
  • Acceptance tests
  • Application code

Sample Stories (from XP Installed)

Union dues vary by union, and are taken only in the first pay period of the month. The system computes the deduction automatically. The amount is shown in the attached table.
... Sometimes we need to split a story

Allow the user to add new service types to the system’s initial list. For example, he may wish to add a special entry for getting the car washed at the high school’s “free” wash. Include the standard fields amount and date, plus allow the user to add any additional text or numeric fields. Reports should automatically sum any numeric fields. (Programmer note: story needs splitting. Please separate text and numeric fields into two stories, plus one for the summing.)

(Split 1) Allow the user to add new service types, including the standard fields plus any additional text fields desired.

(Split 2) Allow the user to add numeric fields to user defined service types.

(Split 3) In all reports, show totals of all numeric fields, not just the standard gallons and dollar amount fields.