System Analysis

Software Engineering

2004-2005

Marco Scotto (Marco.Scotto@unibz.it)
Content

Introduction

- Goals, human actors, & 3 beasts
- Traditional approach
- Diagrams
- Issues & drawbacks
Introduction

- Impossible to provide a precise definition
- A phase between requirement elicitation and system design
  - Purposes of the system are formalized and put in a consistent and coherent framework
- Traditional approach
Content

- Introduction
- Goals, human actors, & 3 beasts
  - Traditional approach
  - Diagrams
  - Issues & drawbacks
## Goals, human actors, & 3 beasts

### Goals

- Understand **requirements**, resolve ambiguities and incompleteness
- Lay out **basic model** of system
- Understand **what is needed** for development

### Human actors

- **Customer** – a stakeholder: orders and pays the system
- **Manager** – a stakeholder: heads and controls the development team
- **Developer**: builds the system
- **Analyst** – specialized developer in system analysis

### 3 beasts

- **Uncertainty**: customer does not know the requirements, or they are ambiguous, incomplete and unstable
- **Irreversibility**: Once a basic model is decided, changes are costly
- **Complexity**: Requirements and/or basic model too complex
Content

- Introduction
- Goals, human actors, & 3 beasts
- Traditional approach
- Diagrams
- Issues & drawbacks
Traditional approach (1/5)

Plan-driven approach to analysis:
- Try to be as specific as possible, resolve up-front all ambiguities, build a complete and consistent set of formal specifications, and develop a solid base on which to build the system
Traditional approach (2/5)

The Specification Document

- **Goal:** reports unambiguously the system requirements
- **Key contractual document:** the customer approves and signs it
- **Seldom only textual**
- **Data Dictionary**
  - Key part of the document
  - Repository containing the definition of all the data and control info entities in input or output to the various modules of the system
Traditional approach (3/5)

Goal: modeling the system

- 3 Aspects to model
  - Data structure
  - Functionalities
  - Behaviour

- Using various types of diagrams & notations
  - Based on info flow, procedures (describe behaviour)
  - Based on database field (describe data structures)
Traditional approach (4/5)

➢ Most popular diagrams used in “structured analysis”

- **Data Flow Diagram**
  - Captures the flow of info and control
- **State Diagram**
  - Describes the possible states of the system and the admissible state changes
- **Entity-Relationship Diagram**
  - Describes the data structure of a database
Traditional approach (5/5)

- Formal specification techniques
  - Goal: describes the system “mathematically”
  - Separating “what” from “how”
  - Providing a complete mathematical (formal) specification of the system
    - using proper notation and languages
  - Ideally, the correctness of the system could be mathematically proven
Content

- Introduction
- Goals, human actors, & 3 beasts
- Traditional approach

- Diagrams
- Issues & drawbacks
Data flow diagrams – DFD (1/3)

- 2 purposes
  - Define data flow and transformation
  - Specify functions which process and transform data

- Major drawback: difficult to translate DFD into system architecture and into code

- Drawing DFD for complex system adds irreversibility and complexity
Data flow diagrams – DFD (2/3)

Basic entities

- **External entity:**
  - produces info to be fed into system
  - can be a person or another program

- **Transformation process**
  - Represents a system’s activity
    - receives input \( \rightarrow \) processes & transforms \( \rightarrow \) produces output

- **Data store**
  - file or database table (permanent possible)

- **Data flow**
  - Refers to info between DFD entities
  - plain data or control info

![Diagram](image)
Data flow diagrams – DFD (3/3)

Example of an automated auction system

1. Control and register bid
   - Bid placed
   - Bid and user data
   - Bid accepted message
   - Wrong bid message
   - Rejected user message

2. Accept or reject user
   - Bid data
   - Bid accepted message

3. Accept bid and update Web site
   - D3 Web site

4. Generate daily reports
   - Summary report
   - Detail list
   - D1 Bids DB
   - D2 Users DB

Software Engineering
State diagrams (1/3)

- Software systems as finite state machines
- Infeasible to describe entire project with state diagrams
  - but practical when project is broken into subsystems
- UML state diagram symbols:
State diagrams (2/3)

➢ State in a box
  • Lower portion holds listing of internal actions when object remains in the state

➢ Substates
  • represented in composite state
  • Can be concurrent and/or sequential
  • Fork used when entering composite state
  • Join used when leaving composite state

➢ State diagrams
  • useful for describing behaviour of parts of a system
  • Part of object oriented analysis and design
State diagrams (3/3)

- Item inserted into the system:
  - Waiting
  - On display
  - On auction
  - Not sold
  - Sold
  - Archived
- Transition arrows:
  - Put on display:
  - No more displayed:
  - Put on auction:
  - Valid bid received:
  - End of bidding with no valid offer:
  - Sale cancelled:
  - End of bidding with a valid offer:
  - Delivered to buyer:

Example: state of an item on sale in online auction management system
Entity-relationship diagrams (1/2)

- Data modeling: specification of data processed
- Main goal – finding and defining:
  - Primary data objects in terms attributes
  - Relationships among data objects
  - Constraints on the data structure
- Main notation: Entity-Relationship Diagram (ERD)
  - Focuses solely on data
  - Mainly for analysis and design of database of system
- DFD and State diagrams: dynamic views
- ERD diagrams: static views
Entity-relationship diagrams (2/2)

- **Entity**: data object
  - Composed of and described by attributes
    - Attribute is a data item simple enough to be considered an info unit
  - Distinction between entities and attributes depends on the abstraction level of developers

- **Relationship**: link among entities
  - Can have attributes

```
Entity with attributes

Student

student code
student name
registration date

Link between entity and relationship

Relationship

Symbols of ERD
```
Cardinality and modality (1/2)

- **Cardinality = multiplicity**
  - Number of possible occurrences of one entity that can be related to the number of occurrences of the other entity
  - Its value can be either one or many

- **Modality = necessity of participation of one entity in a relationship**
  - Either optional or mandatory

![Diagram](Image)
Cardinality and modality (2/2)

Example:

• A driver can drive \( \geq 0 \) car
• A car has only one driver
• A car parks at one or more parking spots
• A parking spot is used by one or more cars
Content

- Introduction
- Goals, human actors, & 3 beasts
- Traditional approach
- Diagrams
- Issues & drawbacks
Issues & drawbacks

- Enormous specification documents
  - Increases irreversibility and complexity
- Analysis-paralysis: never-ending analysis phase
  - Uncertainty caused by requirement changes
- Loss of customer’s interest and support
- Partial solution: incremental (small parts of system) and iterative (subsets of features) approach
- Alternative approach: OO analysis