Integrative Systems Engineering
System Modeling for Requirements Engineering

Prof. Michael Zink
Overview

- Introduction
- Representation for Requirements Engineering
  - Data Flow Diagrams
  - Entity-Relationship Diagrams
  - Statecharts
- Methods
  - Viewpoint Methods
    - Controlled Requirements Expression (CORE)
    - Structured Analysis and Design Technique (SADT)
    - Viewpoint-oriented Requirements Definition (VORD)
  - Formal Methods
Introduction

- Modeling supports analysis and design process by introducing formality into systems definition
- Often pictures are used to visualize aspects of development
- Modeling provides a way of formalizing these representations through diagrams
- Serves also as a medium to understand and communicate system development ideas
- Art of modeling is most creative aspect of System Engineer’s work
Introduction

- Modeling has the following benefits:
  - Encourage the use of *precisely defined vocabulary* consistent across system
  - Allow system specification and design to be *visualized in diagrams*
  - Allows consideration of *multiple interacting aspects* and views of a system
  - Supports the *analysis of systems* through a defined discipline
  - Allows *validation* of some aspects of system design through animation
  - Allows *progressive refinement* toward detailed design, permitting *test case generation* and *code generation*
  - Encourages *communication between different organizations* by using common standard notations
Flashback: Requirements and Modeling

- Important to understand relationship between requirements management and system modeling
  - Mutually supportive activities that should not be equated
  - Relationship almost like a sandwich:
    - Requirements management: *Bread & Butter*
    - Modeling: *Filling*
- There is no such thing as *requirements modeling*
- Assists engineer in decomposing requirements
Representation for Requirements Engineering

- Data Flow Diagrams (DFDs)
- Entity-Relationship Diagrams
- Statecharts
- Object Oriented Approaches
  - Class Diagrams
  - Use Cases
Data Flow Diagrams

- Basis of most traditional modeling methods
- Minimalist graphical representation of system structure and interfaces
- Can be used to show any type of flow
- DFDs do NOT show control flow
- Elements in DFD are:
  - Data flows (labeled arrows)
  - Data transformations (circles or “bubbles”)
  - Data stores (horizontal parallel lines)
  - External entities (rectangles)
Data Flow Diagrams

- Context diagram is top-level of DFD
- Shows external system interacting with the proposed system
Data Flow Diagrams

- Bubbles can be decomposed another layer down
Data Flow Diagrams

Context Diagram for Ambulance Command & Control

- Caller
- Ambulance
- Records

Other potential external entities:
- Police
- Fire brigade
- Other ambulance C&C systems
Internal system functionality

- Handle callers
- Handle ambulances
- Current incidents
- Keep records
- Ambulance states
- Records
- Caller

Data Flow Diagrams
Data Flow Diagrams

Detailed Model

Handle Callers
  - Communicate with caller
    - Obtain incident details
    - Provide on-line advice
    - Analyze incident
      - Keep records
      - Monitor incidents
        - Monitor ambulance state
          - Allocate ambulances
            - Handle Ambulances
              - Communicate with ambulances
                - Handle Callers

Ambulance

Records

Ambulance states

ECE Department
Data Flow Diagrams: Functional Structure

C&C System

Handle Callers
- Communicate with caller
- Obtain incident details
- Analyze incident
- Provide on-line advice

Handle Ambulances
- Communicate with ambulance
- Allocate ambulance
- Monitor ambulance states

Keep Records
- Monitor incidents
- Provide statistics

The “obtain incident details” function shall allow the call center staff to obtain and record incident details from the caller.

The “allocate ambulance” function shall allow the controller to allocate an ambulance to an incident.
Data Flow Diagrams: Functional Structure

C&C System

- Handle Callers
  - Communicate with caller
  - Obtain incident details
  - Analyze incident
  - Provide on-line advice

- Handle Ambulances
  - Allocate ambulance
  - Communicate with ambulance
  - Monitor ambulance states

- Keep Records
  - Monitor incidents
  - Provide statistics

Performance end-to-end time < 15 seconds

An input here...
Data Flows

System transactions

Handle Ambulances

Allocate ambulances

Communicate with ambulances

Monitor ambulance state

Ambulance

Communicate with caller

Obtain incident details

Analyze incident

Current incidents

Provide on-line advice

Handle Callers

Caller

Keep records

Monitor incidents

Provide statistics

Records

Monitor ambulance state

Ambulance states

Handle Callers

Communicate with caller

Obtain incident details

Analyze incident

Provide on-line advice

System transactions

Handle Ambulances

Allocate ambulances

Communicate with ambulances

Monitor ambulance state

Ambulance

Communicate with caller

Obtain incident details

Analyze incident

Provide on-line advice

Handle Callers

Caller

Keep records

Monitor incidents

Provide statistics

Records

Monitor ambulance state

Ambulance states

Handle Ambulances

Allocate ambulances

Communicate with ambulances

Monitor ambulance state

Ambulance

Communicate with caller

Obtain incident details

Analyze incident

Provide on-line advice

Handle Callers

Caller

Keep records

Monitor incidents

Provide statistics

Records

Monitor ambulance state

Ambulance states

Handle Ambulances

Allocate ambulances

Communicate with ambulances

Monitor ambulance state

Ambulance

Communicate with caller

Obtain incident details

Analyze incident

Provide on-line advice

Handle Callers

Caller

Keep records

Monitor incidents

Provide statistics

Records

Monitor ambulance state

Ambulance states

Handle Ambulances

Allocate ambulances

Communicate with ambulances

Monitor ambulance state

Ambulance

Communicate with caller

Obtain incident details

Analyze incident

Provide on-line advice

Handle Callers

Caller

Keep records

Monitor incidents

Provide statistics

Records

Monitor ambulance state

Ambulance states
Data Flows

- In summary DFDs:
  - Show overall functional structure and flows
  - Identify functions, flows and data stores
  - Identify interfaces between functions
  - Provide a framework for deriving system requirements
  - Tools are available
  - Are widely used in software development
  - Are applicable to systems in general
Entity-Relationship Diagrams

- Provide a means of modeling the entities of interest and the relationships that exist between them
  - *Entity*: Object that can be distinctly identified
    - customer, supplier, part, product
  - *Property*: Information that describes entity
  - *Relationship*: Expresses nature of association
    - one-to-one, one-to-many, many-to-many
  - *Subtype*: Subset of another entity
- Model is independent of processing which is required to generate or use information
Entity-Relationship Diagrams

- **Person** (0..N) involves **Incident** (1..1)
  - **Crew member** (0..1) is a
    - **Crew** (1..N) consists of
      - **Ambulance** (0..N) is staffed by
  - **Hospital** (1..N) is allocated
    - **Allocation** (0..1) has resource
      - **Incident** (0..N) is allocated
        - **Crew member** (0..1) is allocated
          - **Person** (1..1) is allocated
Statecharts

Aircraft

Able to taxi

Taxiing

On stand

Taxi-in

Taxi-out

Cleared for take-off

On Runway

Ready for take-off

Proceed

Taking off

Abort

Braking

Leave runway

Cruising

Descending

Airborne

Ascending

In flight

Cleared to land

Abort

Wheels off

Touchdown

Proceed
Object-oriented Approaches

- We talked about this extensively during the SysML lecture
Methods

- A method is a degree more descriptive than a modeling approach:
  - It tell us what to do and in what order to do it
- Method use various representations e.g.,
  - Natural language
  - Diagrammatic forms
  - Formal mathematics
- Purpose of representations used in methods is to capture information
- Information capture is aided by defining set of concepts that diagram represents
Viewpoint Methods

- Recognizes that requirements should not be considered from a single perspective
- Build on premise that requirements should be collected and organized from different viewpoints
  - Viewpoints associated with stakeholders
  - Viewpoints associated with organizational and domain knowledge
- Three different methods based on viewpoints:
  - Controlled Requirements Expression (CORE)
  - Structured Analysis and Design Technique (SADT)
  - Viewpoint-oriented Requirements Definition (VORD)
Controlled Requirements Expression (CORE)

- Developed following on requirements analysis for UK Ministry of Defense
- Finding: Methods often started by defining context of a solution to a problem before assessing possible solutions
- Central concept: Viewpoint and associated representation known as viewpoint hierarchy
- Person, role, or organization can be a viewpoint
- Viewpoint can also represent intended system and its subsystems
Controlled Requirements Expression (CORE)

- Viewpoint hierarchy
- Tabular collection from (TCF)
- System transaction
- Single viewpoint model (SVM)

Structures

- Viewpoint
  - Flow
    - Inter-viewpoint flow
    - Internal flow
  - Can be produced by
  - Can be used by
  - Can trigger
  - Can control

Function

Data store

Can change
Example: Aircraft Brake & Control System

- Pilots
- Aircraft
- Braking System x 2
- Steering System
- Maintenance engineer
- Cockpit
- Sensors
- Brake pedals
- Environment
- System recording
# Tabular Collection Form

<table>
<thead>
<tr>
<th>Source</th>
<th>Input</th>
<th>Action</th>
<th>Output</th>
<th>Dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel 1,2</td>
<td>Power On of channel 1, 2</td>
<td></td>
<td>Self test OK</td>
<td>Channel 1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Self test fail</td>
<td></td>
</tr>
<tr>
<td>Cockpit</td>
<td>Power up</td>
<td>Power up self test</td>
<td>Channel fault</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NWS Isolator Valve Fault</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Autobrake Fault</td>
<td>System Recording</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Shutoff Valve Fault</td>
<td></td>
</tr>
<tr>
<td>Other Sensors/Actuators</td>
<td>Towing Controlled</td>
<td>Monitor Towing</td>
<td>Towing Control On</td>
<td>Aircraft</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Towing Control Off</td>
<td></td>
</tr>
<tr>
<td>Channel 1,2</td>
<td>Operational of Chan. 1,2</td>
<td></td>
<td>Towing Control Off</td>
<td></td>
</tr>
<tr>
<td>Wheel Speed</td>
<td>Wheel Speeds</td>
<td>Monitor Wheel Speeds</td>
<td>Speed &gt; 70 Knots</td>
<td>Cockpit</td>
</tr>
</tbody>
</table>

ECE Department
Structured Analysis and Design Technique

- Purely hierarchical approach to the problem with succession of blueprints
  - *Modularizing* and
  - *Refining* it until solution is achieved
Structured Analysis and Design Technique

- **Elements:**
  - Box, represents activity (in activity diagram) or data (in data diagram)
  - Arrows, that join boxes and represent data needed or provided by activities
    - *Input* arrows enter box from left side, and represent data that is available to activity
    - *Output* arrows exit box from right side, and represent data that is produced by activity
    - *Control* arrows enter box from top, and govern way in which transformation is taking place
    - *Mechanism* arrows enter box from bottom, and control way in which activity may use outside mechanisms
SADT

Decomposition

System top level

2nd level

3rd level
Viewpoint-oriented Requirements Definition

- Method based on viewpoints
- Service-oriented model where viewpoints are to be considered clients (almost like client-server system)
- Viewpoint:
  - Receives services from system
  - Passes control information to the system
- Well-suited for specifying interactive systems
- Two types of viewpoints:
  - **Direct viewpoints** receive services from system and send control information and data to system
  - **Indirect viewpoints** do not interact directly with system but rather have an “interest” in some or all services delivered
Viewpoint-oriented Requirements Definition

- Three main iterative steps in VORD:
  - Viewpoint identification and structuring
  - Viewpoint documentation
  - Viewpoint requirements analysis and specification

- Viewpoints are represented by rectangles:
  - Identifier
  - Label
  - Type

- VORD methods guides System Engineer in identifying viewpoints.

- Provides abstract viewpoints as starting point
VORD

Viewpoint identifier

Label

Attribute identifier

Type

n

m

attribute

n.1

n.2
1. Operator
   Car Park Customer

2. Operator
   Car Park Staff

1.1 Operator/Customer
   Organization
   Parking Company

1.2 Operator/Customer
   System
   Credit Card D/B

2.1 Operator/Staff
   System
   Ticket Issuing System

2.2 Car Park Manager
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Label</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer</td>
<td>1.1  Provide facility for ticket based on suitable payment and length of stay</td>
<td>sv</td>
</tr>
<tr>
<td>1.1</td>
<td>Credit Card User</td>
<td>1.1.1 Provide facility based on valid credit card</td>
<td>sv</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.2 Provide ticket issuing service for customer</td>
<td>sv</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.3 Ticket issuing service should be available 99/100 requests</td>
<td>nf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.4 Ticket issuing service should have a response time of no more than 30 seconds</td>
<td>nf</td>
</tr>
<tr>
<td>1.2</td>
<td>Cash User</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Formal Methods

- More rigorous representation based on mathematics
- Can be used to:
  - Conduct mathematical proof of consistency of specification and implementation
  - Rigorous checking to eliminate some kinds of errors
- Necessary for certain systems like:
  - Nuclear power plants
  - Weapons and aircraft control systems
- Most common methods for formal definition of functionality:
  - Z
  - VDM
  - Lotos and B
Z - A Model-based Formal Method

- Based on first-order predicate logic and set theory
- Data can be represented as sets, mappings, tuples, relations, sequences, and Cartesian products
- Schema:
  - Signature part is list of variable declarations
  - Predicate part consists of single predicate
- Specifications are represented as collections of schemas.
- “?” denotes variable as input of operation
- “!” denotes variable as output of operation
**Z - Method**

**Library == [shelved; P Book: readers: P Reader: stock; P Book: issued: P Book]**

**Issue**

\[ \Delta \text{Library} \]

\[ b? : \text{Book} \]

\[ r? : \text{Reader} \]

\[ b? \in \text{shelved}; r? \in \text{readers} \]

\[ \text{Issued} = \text{issued} \oplus \{b?-r?\} \]

\[ \text{Shelved} = \text{shelved}\backslash\{b?\} \]

\[ \text{Stock} = \text{stock: reader} = \text{readers} \]