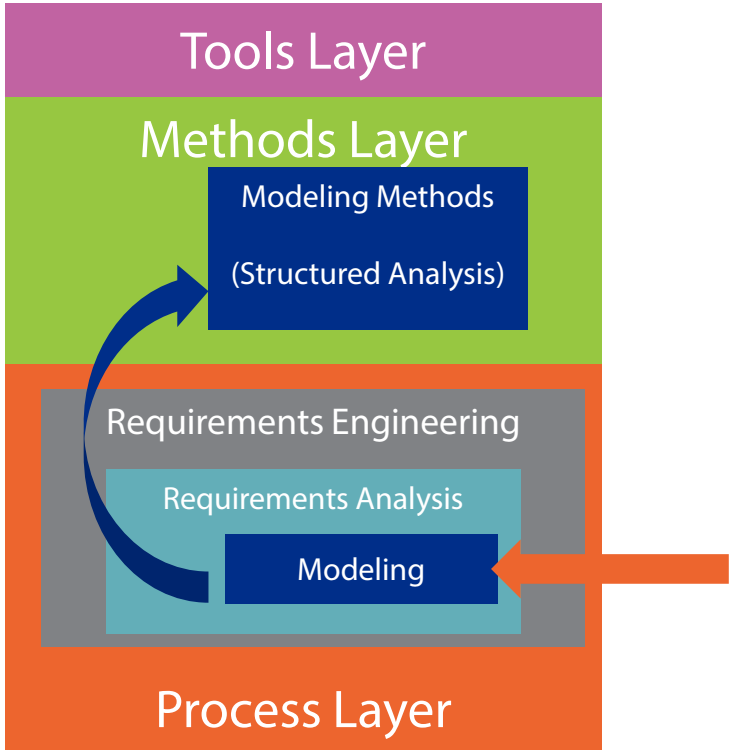
# Requirements Modeling – Structured Analysis

Mohamad Halabi
@mohamadhalabi

pluralsight
hardcore dev and IT training

# Introduction



# Analysis Modeling

- **Analysis modeling is about the "what" not the "how"**
  - The "how" is the focus of the Design phase
- **Analysis modeling answers important questions, such as:**
  - How actors interact with the system?
  - What information does the system consume and produce?
  - What functions does the system perform?
  - How does the system behave in response to certain events?
  - Under what constraints does the system operate?
- **Analysis models serve various purposes:**
  - Show the system at different abstraction levels
  - Become the main driver of the Design phase
  - Become input to derive Acceptance Tests

# Modeling Rules

1. **Strive for simple models**
   a) Models have a purpose; being fancy is not one of them!
   b) Models are not useful if they cannot be understood by stakeholders
   c) Models are also not useful if they cannot be understood by designers
2. **Any useful model is a good model**
   a) Strictly following an analysis method is not mandatory
   b) If it brings value, you can use notations from various methods
   c) Models have purpose. Any notation that helps is a good notation

# Modeling Methods

## Structured Analysis

Mature; has been used for long time

Models the system in terms of data objects and flow of data within the system

## Object-Oriented Analysis

More recent approach

Models the system in terms of objects and their interaction

# Structured Analysis

- **Four models are created: data, functional, information-flow and behavioral**
- **Data model: Entity Relationship Diagram (ERD)**
  - Data objects, attributes, and relationships
- **Functional and Information-Flow models: Data Flow Diagram (DFD)**
  - How data objects are transformed as they flow in the system
  - The functions that perform the transformation
- **Behavioral model: State Transition Diagram (STD)**
  - How system changes state in response to events

# Data Modeling

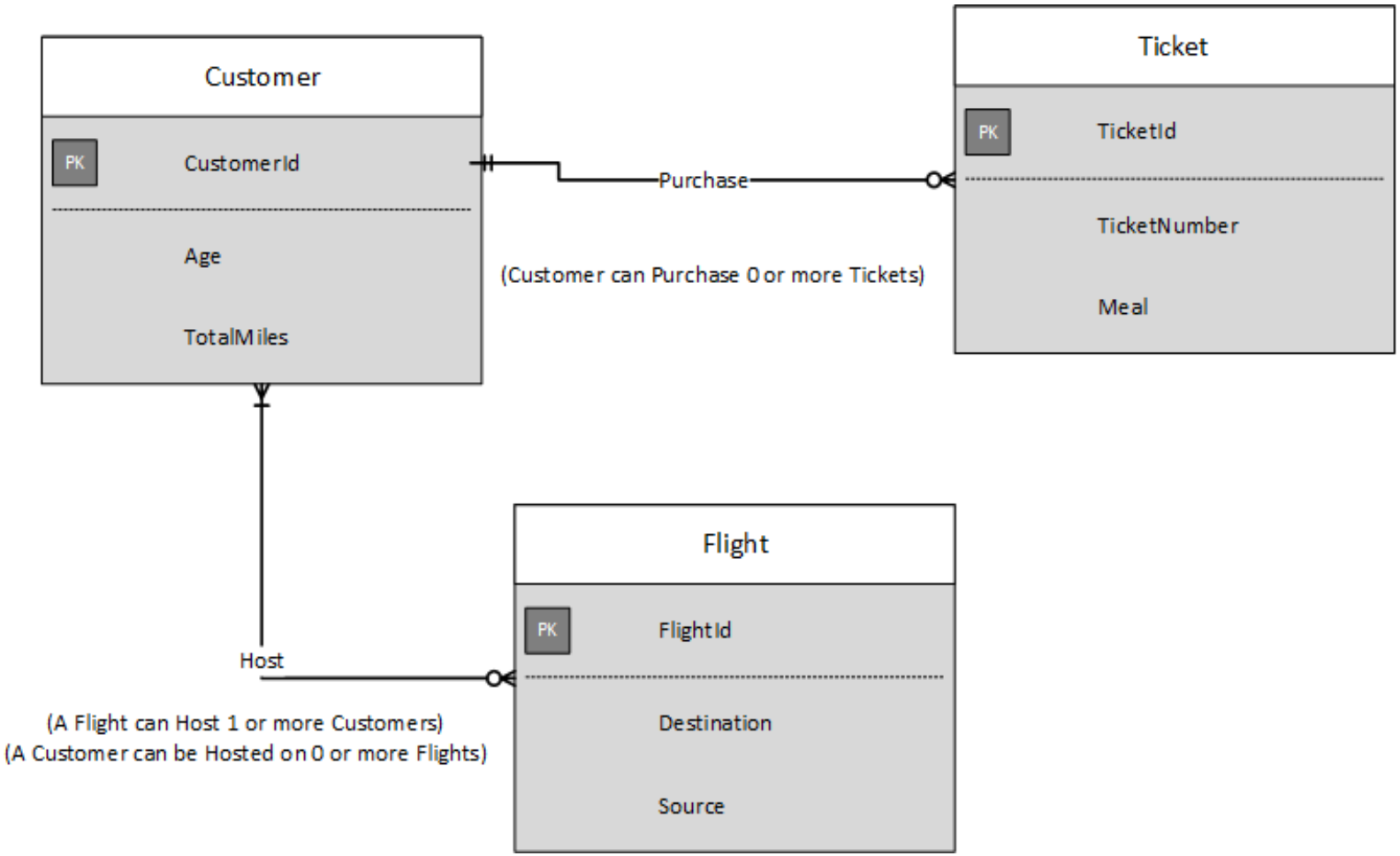
- **Data modeling specifies:**
  1. Data objects consumed and produced by the system
  2. Relationships between these objects

# Data Object

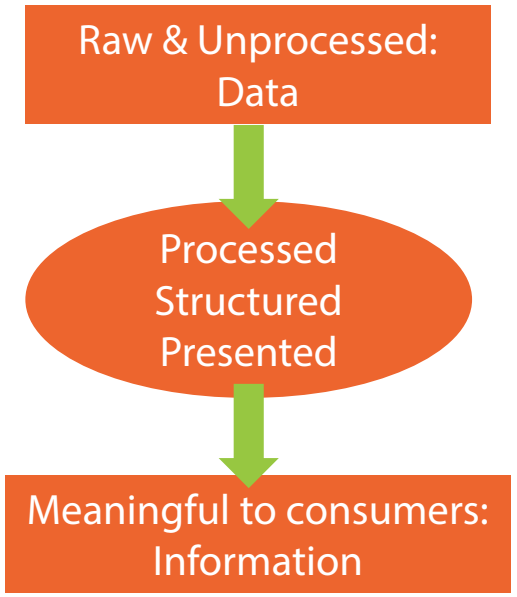
- **Represents information that is produced/consumed by the system**
  - Ex: “Customer” and “Ticket” are data objects of a Ticket Reservation system
- **Has attributes (properties)**
  - Ex: “Age” and “Gender” attributes of “Customer” are consumed by the system
  - Ex: “TotalMiles” attribute of “Customer” is generated by the system
- **Does not specify operations**
  - Operations are part of objects in Object-Oriented analysis
- **Has relationships with other data objects**
  - Ex: “Customer” *purchases* a “Ticket”

# Entity Relationship Diagram (ERD)

- ERD notation models data objects and their relationships



# Data vs. Information




Ex:
- Num of purchased tickers
- Num of complaints

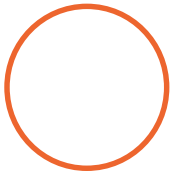
Ex:
90% of complaints are from online channel

- **Systems:**
  1. Accept data from various sources
  2. Data is processed and transformed
  3. Information is produced to customers, users, or other systems

# Data Flow Diagrams (DFD)

- **Serve two purposes:**
  - Shows how data objects flow through the system (Information-flow model)
  - Shows transformation applied on these objects (Functional model)

External entity:
- Supplies or consumes data/information
- It can a system, device, person, etc…

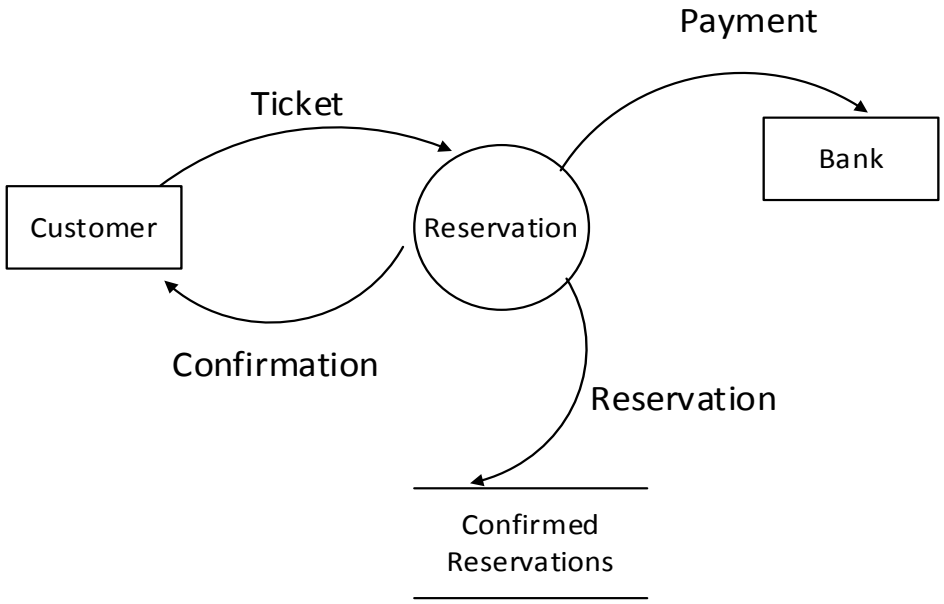


Process or function
- Transforms data objects

Data object

Data store

# DFD Example

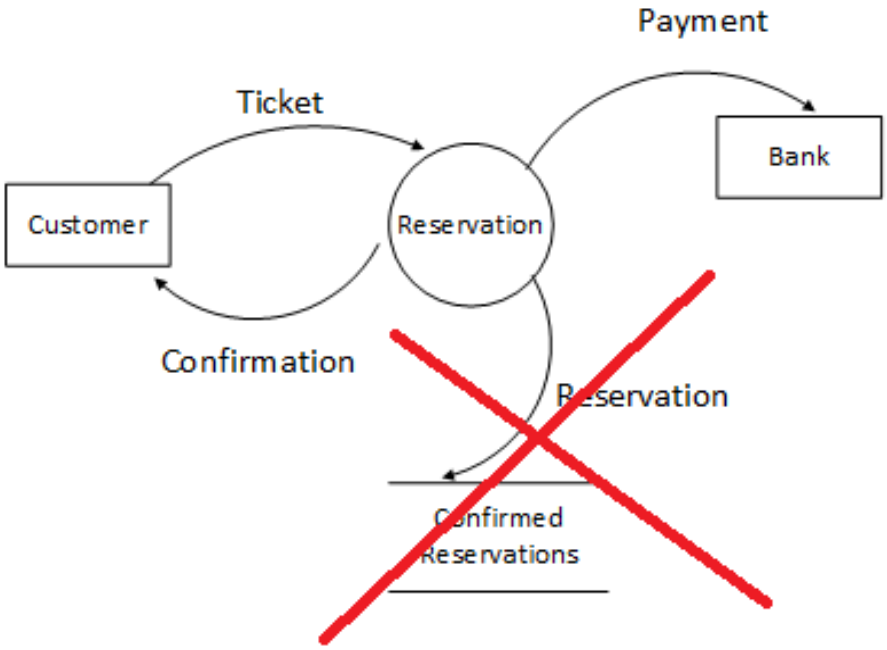

Payment
Ticket
Bank
Customer
Reservation
Confirmation
Reservation
Confirmed
Reservations

# DFD Level

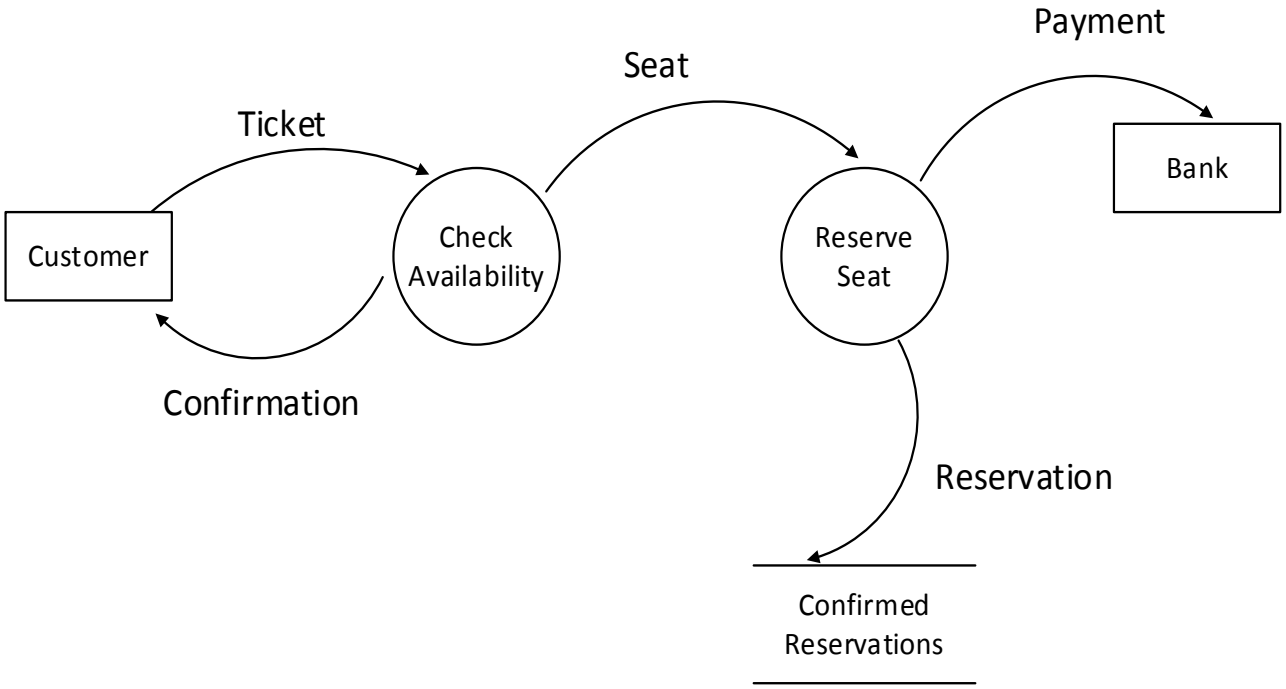
- **DFDs have multiple levels**
- **Each level models a different abstraction level in terms of:**
  - Information-flow
  - Processing detail

# DFD Level-0

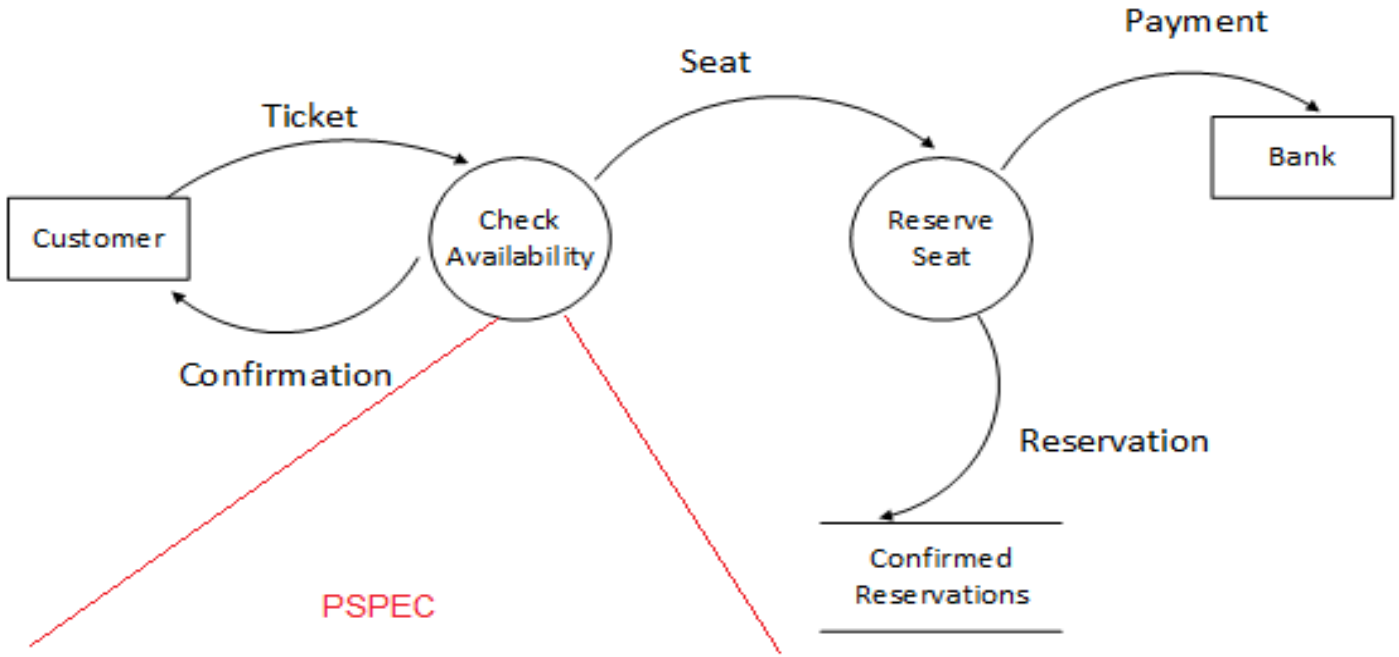
- Called “context model”
- Shows the entire system as a single bubble and the external entities



# DFD Level-2


Payment
Seat
Ticket
Bank
Customer
Check
Availability
Reserve
Seat
Confirmation
Reservation
Confirmed
Reservations

# Process Specification (PSPEC)




*If available seats in flight > requested seats*
*then*
*reserve seat for customer*
*decrease seat availability by number of seats*
*else*
*suggest next available flight*

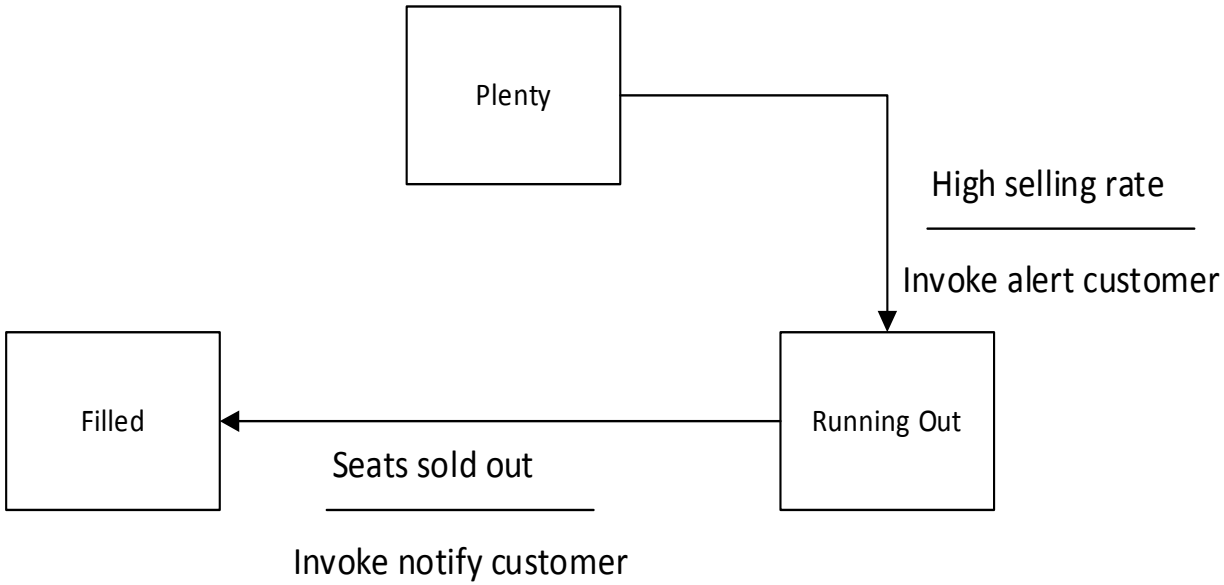
# Behavioral Modeling

- **Behavioral models show how systems change state in response to events**
- **State Transition Diagram (STD) models system's behavior:**
  - System states
  - State changes in response to events
  - Action to be taken as result of the event

# System States

- **Consider the Ticket Reservation system**
  - Customers can initiate a reservation process and continue it later
  - A Notification sub-system monitors the number of seats left
  - Notification sub-system changes its state to: *plenty*, *running out*, or *filled*
  - *Plenty*, *running out*, and *filled* are states that indicate system behavior
  - A State Transition Diagram (STD) moves how the system moves from one of these states to the other

# State Transition Diagram (STD)

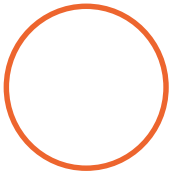


# STD Pre-UML

- **STD existed before UML**
- **However, UML has adapted its own variation of STDs**
  - It calls it Statechart
  - Also called State Machine Diagram or State Diagram
- **STD used in Structured Analysis has different notation than UML State Diagram**

# Control Flow Diagram (CFD)

- CFD shows control flow instead of data flow (DFD)
- CFDs show how events (not data) flow in/out processes
- Events activate/deactivate processes



Process or function



Event

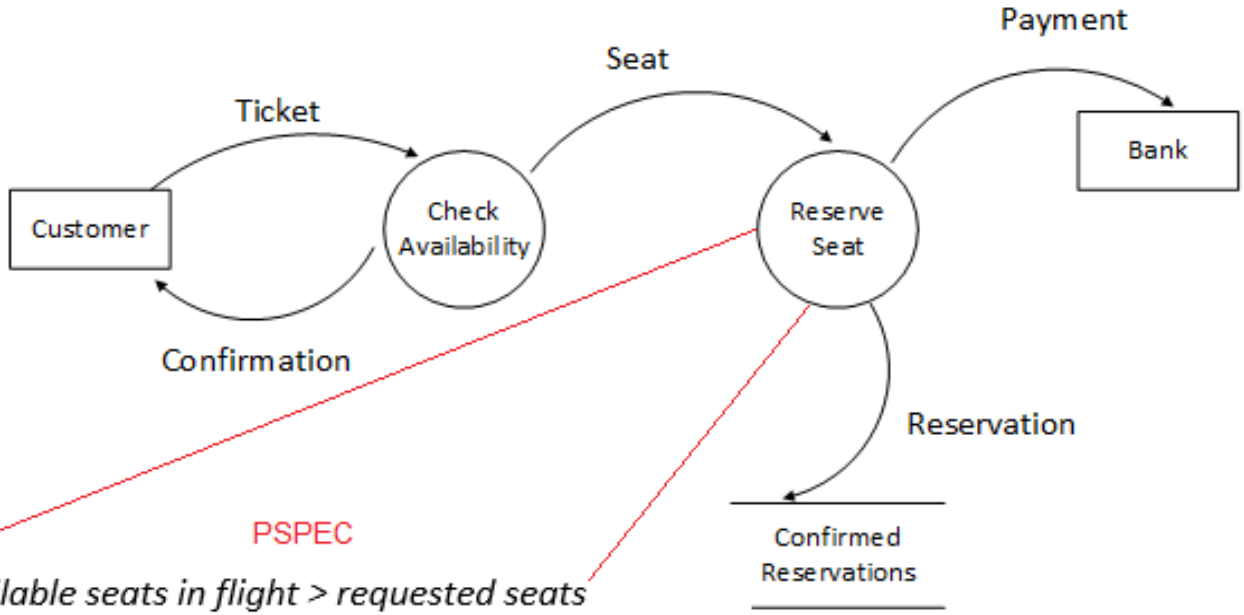


Input/output to/from Control Specification (CSPEC)

- CSPEC describes how control in CFD is handled
- How processes are activate/deactivated
- CSPEC can be described using a STD

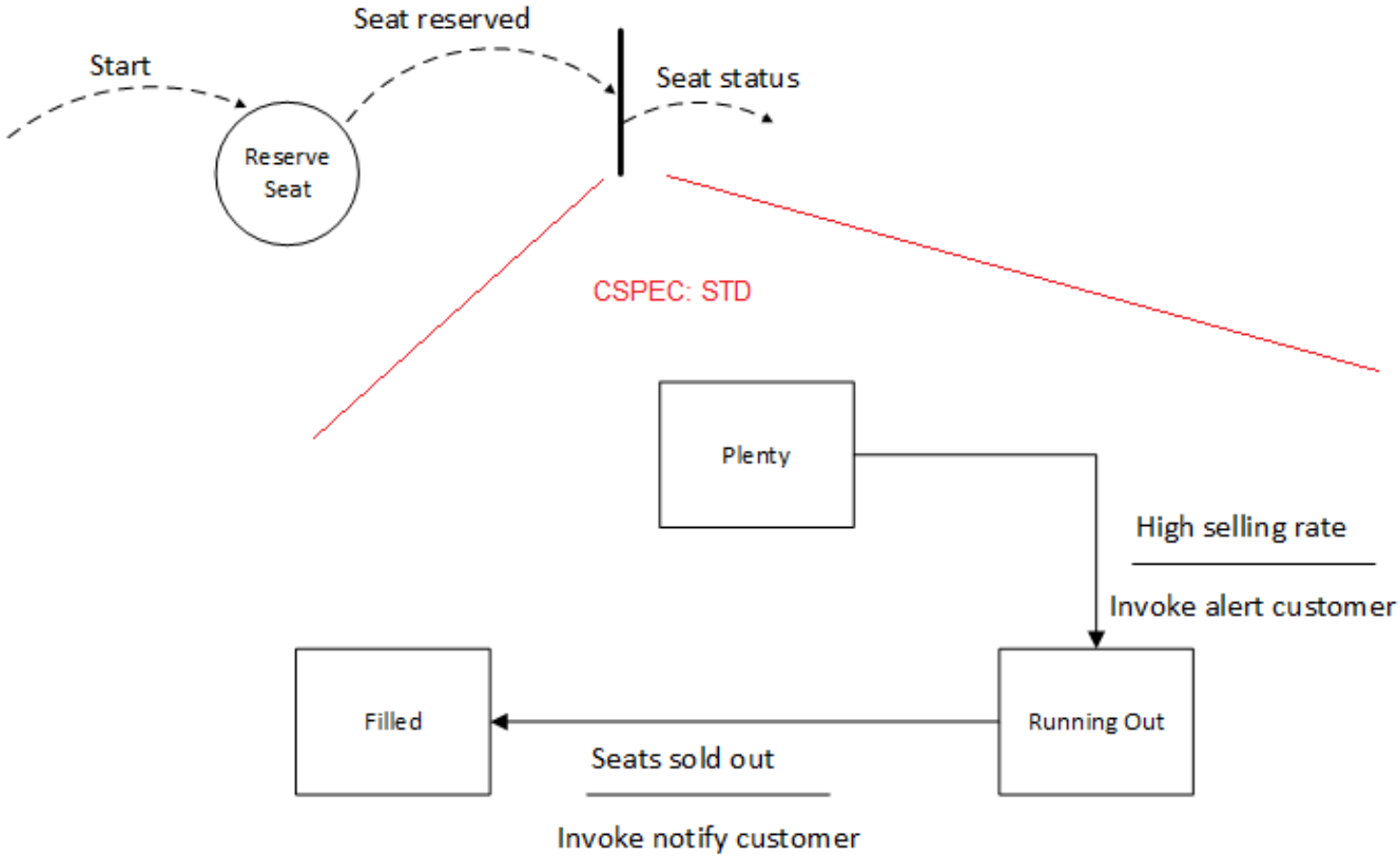
# CFD Example

- **Recall the DFD discussed before:**




*If available seats in flight > requested seats*
*then*
*reserve seat for customer*
*decrease seat availability by number of seats*
*else*
*suggest next available flight*

# CFD Example


Start
Seat reserved
Reserve Seat
Seat status
CSPEC: STD
Plenty
High selling rate
Invoke alert customer
Filled
Running Out
Seats sold out
Invoke notify customer

# Structured Analysis Method in Requirements Analysis

1. **Define data objects and attributes based on Requirements Elicitation information**
2. **Analyze with stakeholders relationships between data objects**
   a) Create ERDs
3. **Model functional and information-flow using DFDs**
   a) DFDs can be supplemented with PSPECs
4. **Model control instead of data using CFDs and CSPECs**
5. **STDs detail CSPECs**
6. **Pass models to design team**
7. **Pass models to testing team**

# See the Importance of Modeling?

- **Making you a Structured Analysis expert was not the intention**
- **Can you see now how models:**
  - Help analysts and stakeholders in system understanding and collaboration
  - Guide the work of implementation team
- **The alternative would be using narrations!**

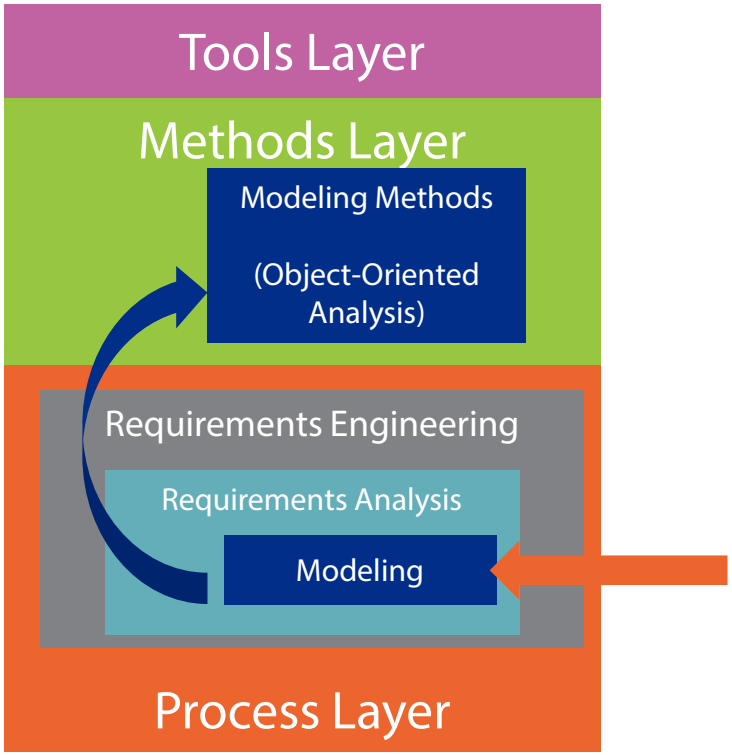
# More Resources

- **“Structured Analysis and System Specification” by Tom DeMarco**
  - http://www.amazon.com/Structured-Analysis-System-Specification-DeMarco/dp/0138543801

# Summary

- **Structured Analysis is an established method for Requirements Analysis**
- **Structured Analysis generate models at different abstraction levels:**
  - ERDs model data objects and their relationships
  - DFDs model functional and information-flow
  - STDs models how the system states change in response to events
  - CFDs and CSPECs model events instead of data

# What's Next?


Tools Layer
Methods Layer
Modeling Methods
(Object-Oriented Analysis)
Requirements Engineering
Requirements Analysis
Modeling
Process Layer