
Chapter 2 - Tutorial

Guided Tutorial in Analysis Modeling
OnLine Shopping

Copyright © 2000 by Addison Wesley
Version 1.0
Topics

- Online Shopping – Tutorial Statement
- Use Case Modeling
- Activity Modeling
- Class Modeling
- Interaction Modeling
- Statechart Modeling
OnLine Shopping – Order Processing

- **Buying computers via Internet**
- **The customer can select a standard configuration or can build a desired configuration online**
- **To place an order, the customer must fill out the shipment and payment information**
- **The customer can check online at any time the order status**
- **The ordered configuration is shipped to the customer together with the invoice**
Use case modeling

- **Use case** - outwardly visible and testable system behavior
- **Actor** - whoever or whatever (person, machine, etc.) that interacts with a use case
- Actor receives a **useful result**
- Use case represents a complete unit of functionality of value to an actor
- There may be some use cases that do not directly interact with actors
- In many instances, a function requirement maps directly to a use case
- **Use Case Diagram** is a visual representation of actors and use cases together with any additional definitions and specifications
- **UML diagram** is synonymous with **UML model**
Consider the requirement:
After customer’s order has been entered into the system, the salesperson sends an electronic request to the warehouse with details of the ordered configuration.
Use cases

- The customer uses the manufacturer’s online shopping Web page to view the standard configuration of the chosen server, desktop or portable computer.

- The customer chooses to view the details of the configuration, perhaps with the intention to buy it as is or to build a more suitable configuration.
Use Case Diagram

The <<extend>> relationship - the use case Order Configured Computer can be extended by Customer with the use case Request Salesperson Contact.
Documenting use cases

- **Brief Description**
- **Actors involved**
- **Preconditions** necessary for the use case to start
- **Detailed Description** of flow of events that includes:
  - **Main Flow** of events, that can be broken down to show:
    - **Subflows** of events (subflows can be further divided into smaller subflows to improve document readability)
  - **Alternative Flows** to define exceptional situations
- **Postconditions** that define the state of the system after the use case ends
### Narrative use case specification

**Brief Description**

This use case allows a Customer to enter a purchase order.

**Actors**

Customer

**Preconditions**

The page displays the details of a configured computer together with its price.

**Main Flow**

The system assigns a unique order number and a customer account number to the purchase order and it stores the order information in the database.

**Alternative Flows**

The Customer activates the Purchase function before providing all mandatory information.

**Postconditions**

If the use case was successful, the purchase order is recorded in the system’s database.
Activity Modeling

- **Activity model**
  - Can graphically represent the flow of events of a use case
  - Can be used to understand a business process at a high-level of abstraction before any use cases are produced

- **Shows the steps of a computation**
  - Each step is a **state** of doing something
  - Execution steps are called **activity states**
  - Depicts which steps are executed in sequence and which can be executed concurrently
  - **Transition** – the flow of control from one activity state to the next

- **Use case descriptions** are (usually) written from an outside actor’s perspective

- **Activity models** take an inside system’s viewpoint
Activities

- **Activity states** can be established from the use case document
- **Activities** should be named from the system’s perspective, not the actor’s viewpoint
- **Activity** takes time to complete
- **Action** is so quick that – on our time scale – it is considered to take no time at all
- **UML** uses the same same graphical symbol for **activity state** and **action state** – rounded rectangle
The system assigns a unique order number and a customer account number to the purchase order and it stores the order information in the database.
**Activity Diagram**

- **Activity Diagram** shows transitions between activities.
- A solid filled circle represents the **initial state**
- The **final state** is shown using so called bull’s eye symbol.
- Transitions can **branch** and **merge** (diamond) – alternative computation **threads**
- Transitions can **fork** and **re-join** (bar line) – concurrent (parallel) computation **threads**
- Activity diagram without concurrent processes resembles a conventional **flowchart**
Activity Diagram

Multiple exit transitions (branch condition that is internal to activity state)

Explicit branch condition (that appears on exit from activity state)
Class Modeling

- Captures **system state** – the function of the system's information content at a point in time

- Class modeling elements
  - classes themselves
  - attributes and operations of classes
  - Relationships – associations, aggregation and composition, generalization

- **Class Diagram** – combined visual representation for class modeling elements

- Class modeling and use case modeling are typically conducted in parallel
Classes

- So far, we have used classes to define "business objects"
  - Called **entity classes** (model classes)
  - Represent persistent database objects
- Other classes
  - Classes that define GUI objects (such as screen forms) – **boundary classes** (view classes)
  - classes that control the program's logic – **control classes**
- Boundary and control classes may or may not be addressed in requirements analysis – may be delayed until the system design phase
Classes

Is this a class?

- Is the concept a container for data?
- Does it have separate attributes that will take on different values?
- Would it have many instance objects?
- Is it in the scope of the application domain?
**Classes**

- **The warehouse obtains the invoice from the salesperson and ships the computer to the customer**

---

Is **Salesperson** a class or an attribute of **Order** and **Invoice**?

Do we need **Shipment** class? Is it in the scope?
Attributes

Customer
(from Use Case View)
- customer_name : String
- customer_address : String
- phone_number : String
- email_address : String

Order
- order_number : String
- order_date : Date
- ship_address : String
- order_total : Currency
- order_status : String
- salesman_name : String

Payment
- payment_method : String
- date_received : Date
- amount_received : Currency

ConfigurationItem
- item_type : String
- item_descr : String

Computer
- computer_name : String
- standard_price : Currency

ConfiguredComputer
- computer_name : String
- configured_price : Currency

Invoice
- invoice_number : String
- invoice_date : Date
- invoice_total : Currency
Generalizations

Customer
(from Use Case View)

Order

Payment

ConfigurationItem

Computer

Invoice

ConfiguredComputer

StandardComputer
Interaction modeling

- Captures interactions between objects needed to execute a use case
- Shows the sequencing of events (messages) between collaborating objects
- Used in more advanced stages of requirements analysis, when a basic class model is known, so that the references to objects are backed by the class model
- Two kinds of interaction diagrams
  - **Sequence Diagram** – concentrate on time sequences
  - **Collaboration Diagram** – emphasize object relationships
- Prevailing IS development practice – Sequence Diagrams in requirements analysis and Collaboration Diagrams in system design
Interactions

- **Interaction** – set of messages in some behavior that are exchanged between objects across links

- **Sequence Diagram**
  - Objects - horizontal dimension
  - Message sequence - top to bottom on vertical dimension
  - Each vertical line - the object's lifeline
  - Arrow - message from a calling object (sender) to an operation (method) in the called object (target)
    - Actual argument can be
      - input argument (from the sender to the target)
      - output argument (from the target back to the sender).

```
crs_ref.getCourseName(out crs_name)
```

- Showing the return of control from the target to the sender is not necessary
- **Iteration marker** – an asterisk in front of the message label – indicates iterating over a collection
Interactions

: Customer

aConfWin : ConfigurationWindow

getConf

- getConfItem (out item_rec)

displayComputer(item_recset)

aComp : Computer

: Configuration Item

openNew
Interactions

Build It

Base Price: AU$3399 including Tax (Ex tax Price: AU$2924)

Current specifications and prices should be regularly checked. For component specifications please click the underlined text.

Processor: Intel® Pentium® III Processor 700MHz
Memory: 128MB SDRAM
Cache: 256K On-die L2 Cache
Hard Drive: 20GB Ultra ATA HDD (7200 rpm)
Floppy Drive: 1.44MB 3.5" Diskette Drive
Network Card: Not included
CD/DVD Player: 12X DVD-ROM Drive
Operations

- Examining the interactions can lead to the discovery of operations
  - Each message invokes an operation in the called object
  - The operation has the same name as the message

- Similarly, the presence of a message in a Sequence Diagram stipulates the need for an association in the Class Diagram
Operations

<<boundary>>
ConfigurationWindow

<<constructor>> openNew()
displayComputer(item_recset)

Computer

computer_name : String

<<abstract>> getConf()

ConfigurationItem

item_type : String
item_descr : String

getConflItem(out item_rec)
Sequence Diagram

- Customer
- Configuration Window
- Computer
- Order
- Order Window

- openNew
- acceptConf
- getConf
- prepareForOrder
- submitOrder
- displayOrder
- storeOrder
- linkComputer
- emailOrder
- linkCustomer
- linkPa
Sequence Diagram

- Order
- OrderWindow
- Customer
- Payment

- displayOrder
- storeOrder
- linkCustomer
- linkPayment

(c) Addison Wesley
Chapter 2 - Tutorial
Statechart modeling

- Captures dynamic changes of class states – the life history of the class
- These dynamic changes describe typically the behavior of an object across several use cases
- State of an object – designated by the current values of the object's attributes
- **Statechart Diagram** – a bipartite graph of
  - **states** (rounded rectangles) and
  - **transitions** (arrows) caused by **events**
- The concepts of states and events are the same concepts that we know from Activity Diagrams – the difference is that “the states of the activity graph represent the states of executing the computation, not the states of an ordinary object”
States and transitions

- Objects change values of their attributes but not all such changes cause state transitions
- We construct state models for classes that have interesting state changes, not any state changes
- Statechart Diagram is a model of business rules
  - Business rules are invariable over some periods of time
  - They are relatively independent of particular use cases
States and transitions

- Unpaid
  - partial payment to Partly Paid
  - final payment to Fully Paid

Partly Paid
  - final payment to Fully Paid

Fully Paid

(c) Addison Wesley
Chapter 2 - Tutorial
Statechart Diagram

- Normally attached to a class, but can be attached to other modeling concepts, e.g. a use case
- When attached to a class, the diagram determines how objects of that class react to events
  - Determines – for each object state – what action the object will perform when it receives an event
  - The same object may perform a different action for the same event depending on the object’s state
  - The action’s execution will typically cause a state change
Statechart Diagram

- The complete description of a transition consists of three parts:
  - event (parameters) [guard] / action

- **Action** – short atomic computation that executes when the transition fires
  - can also be associated with a state

- **Activity** – longer computation associated with a state
Statechart Diagram

Pending

New Order

stock not available

stock available[ ship date in future ]

Future Order

stock available[ ship date in future ]

[ canceled ]

stock available[ ship date now ] / configureComputer

Cancelled

Ready to Ship

[ canceled ]

ship[ accepted ]

Filled

(c) Addison Wesley

Chapter 2 - Tutorial