CS 220 Spring 2008

Third Deliverable 05/19/2008: Architectural Design, Design Specification, and Test Cases

Thai Phan C00745873

Table of Contents

1. Architectural Design 3

1.1. Sub-systems 3

1.2. 3-tier Client/Server Architecture 4

1.3. Broadcast Model 4

2. Design Specification 5

2.1. Structural Models 5

2.1.1. Object Models 5

2.1.2. ERA Model 5

2.2. Behavioral Models 6

2.2.1. Data Flow Diagram 6

2.2.2. State Machine Model 6

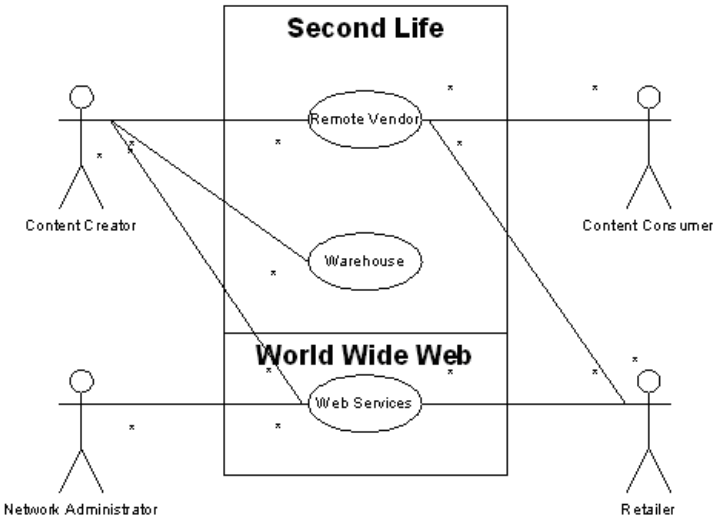
3. Test Cases 7

3.1. Test Cases for Payment Processing 7

3.2. Test Cases for HTTP Request by Remote Vendor 7

3.3. Test Cases for Warehouse HTTP Poller receiving HTTP Response from Web Server 8

1. Architectural Design



In the requirements documentation, we saw that my Vending system was split into three separate use-cases. However, in the actual implementation of the system, the Remote Vendors and Warehouses will be physical entities within Second Life that a user within Second Life can own and maintain. Inside the game, the Remote Vendor will resemble a kiosk or display pedestal, with navigation controls to allow a customer to scroll through a list of merchandise. The Warehouse will resemble a box which the Content Creator can use to store all of his/her merchandise that will be delivered from there.

The Remote Vendor itself will rely on a repository model, since its navigation controls will constantly communicate with the main kiosk.

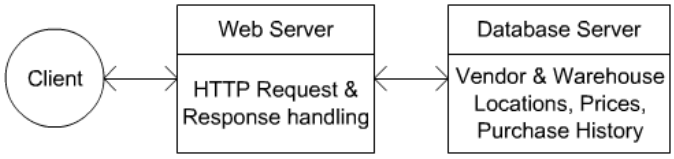
All Remote Vendors and Warehouses are clients which communicate with a Web Server, which stores its data to a database, adopting a three-tier client/server architecture.

Warehouses will check for new purchases made at Remote Vendors by staying in contact with the Web Server (Broadcast Model).

1.1. Sub-systems

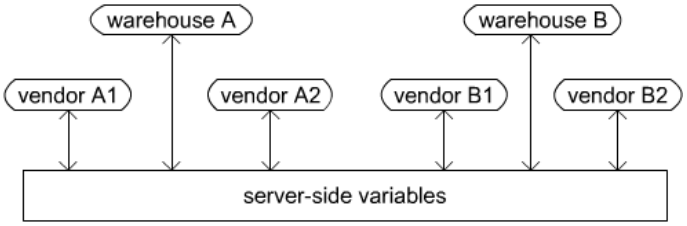
The Remote Vendor is broken down into two modules: a navigation panel and the hologram projector. Because of the way things are implemented inside of Second Life, the navigation panel is broken down into two more component modules, a left arrow button and a right arrow button. All of these components execute their own scripts. The behavior of the navigation panel depends on the list of merchandise, which resides on the hologram projector. Depending on how large this list is, the customer is allowed to loop back to the beginning if he/she has reached the end of the list while using the navigation panel.

1.2. 3-tier Client/Server Architecture



The Remote Vendors and Warehouses are all clients which interact with a remote Web Server. They will communicate with the Web Server through HTTP Requests and Responses. The Web Server will send and retrieve information from a database. The Remote Vendors will retrieve merchandise prices from the database. A Remote Vendor will know which Warehouse to send a purchase request to based on the information on the database.

1.3. Broadcast Model

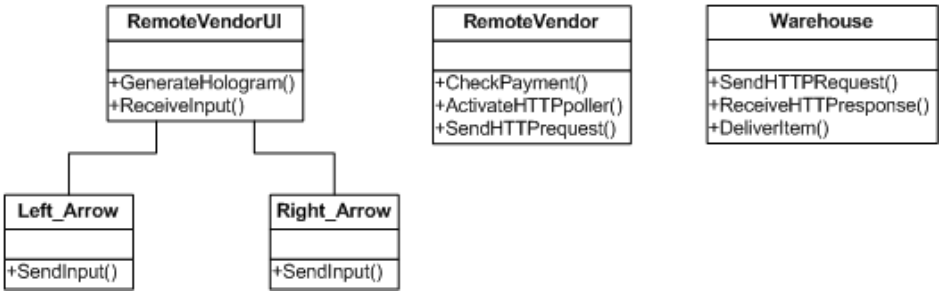


When the Remote Vendor receives payment for a particular item, the Remote Vendor will notify the Web Server, giving the Web Server the key of the customer making the purchase, the name of the item bought, and the key of the Content Creator that made the purchase. The Web Server will then store this information as a Server variable, which can be accessed by any Client. A Warehouse that continuously polls the Web Server (sending HTTP requests to it, and waiting for a response) will discover that a new purchase has been made. If the Warehouse's owner's key matches the key of the Content Creator, then the Warehouse will check and see what item was bought, and will deliver the item of the same name from its Inventory direct to the customer with the matching customer key.

2. Design Specification

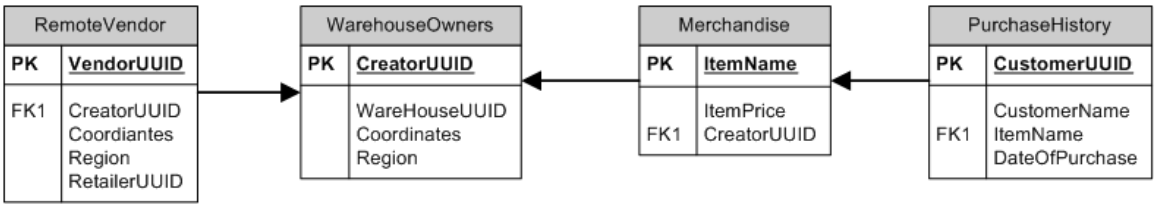
2.1. Structural Models

2.1.1. Object Models



Although Second Life's proprietary scripting language is not object-orientated, the individual scripts that are required to make a single application run behave like individual objects. For example, for a user interface within Second Life to function, it requires scripts for the controls, and a separate script to process the input. This is analogous to having control objects send messages to event handlers inside of a main block of code.

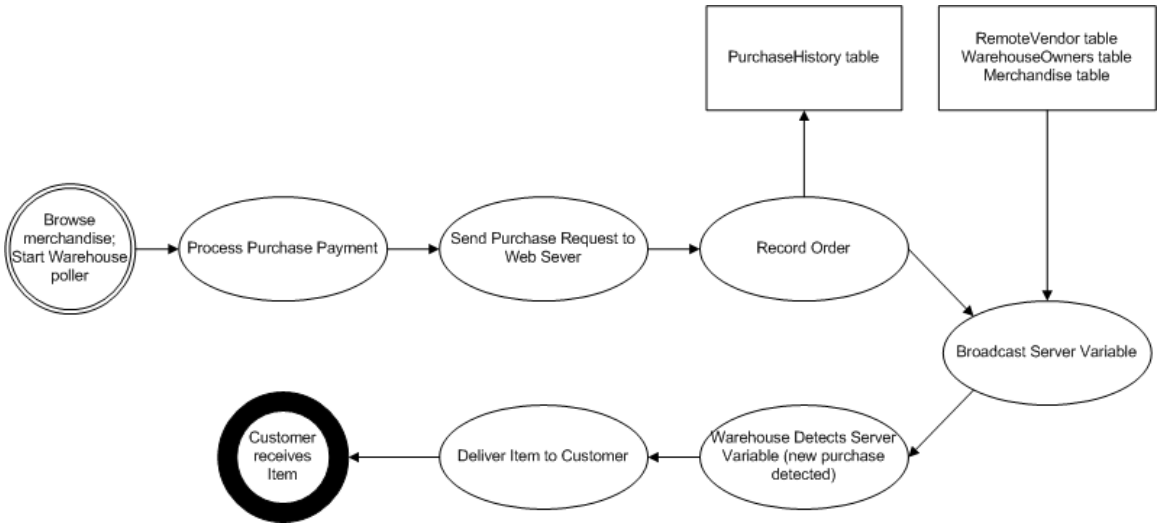
2.1.2. ERA Model



A Content Creator can only have one Warehouse in existence in Second Life. So it is appropriate to record that each Content Creator will have exactly one Warehouse in the table *WarehouseOwners*. A Content Creator can have more than one item stored inside the Warehouse, which is recorded in the *Merchandise* table. A Content Creator will be able to distribute as many Remote Vendors as she wants to as many Retailers as she wants. A Retailer can be in possession of more than one Remote Vendor. Also, there is a table for storing purchase transactions. A Customer can purchase more than one Item.

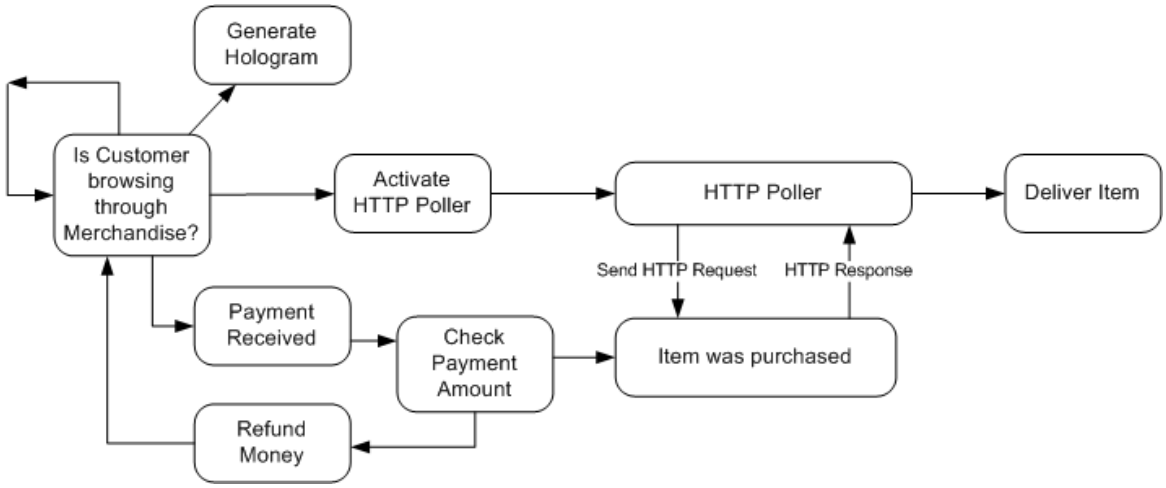
2.2. Behavioral Models

2.2.1. Data Flow Diagram



When a customer in Second Life begins to browse through the list of items for sale at a particular Remote Vendor, the Remote Vendor will tell its corresponding Warehouse to begin polling the Web Server, in order to anticipate a possible new purchase request. When a customer finds the item she wants to purchase, she pays the Remote Vendor. The Remote Vendor will process the amount, and if it is valid, the Remote Vendor will contact the Web Server so that it can record the transaction to the database and determine what item from the corresponding warehouse needs to be delivered to the customer. The server broadcasts this message as a server variable, which the polling Warehouse should detect. Once the Warehouse detects the new purchase request, it will then send the purchased item from its inventory directly to the customer in Second Life.

2.2.2. State Machine Model



3. Test Cases

3.1. Test Cases for Payment Processing

test input	result	input	expected input	output
Pay an amount less than the purchase price	Money is refunded.	$99	$100	You receive $99.
Pay the purchase price	Item is delivered.	$100	$100	You receive item.
Pay an amount greater than the purchase price	Item is delivered. Give change to customer.	$101	$100	You receive item. You receive $1.

3.2. Test Cases for HTTP Request by Remote Vendor

An HTTP request is sent to the Web Server is formatted as:
The name of the item followed by the UUID of the customer making the purchase followed by UUID of the Content Creator who created the item being purchased. Separate each parameter with ++ pattern.

test input	result	input	expected input	output
Send a properly formatted string	Item name is recognized, UUID of customer is valid, UUID of Content creator is valid	Ring++66864f3c-e095-d9c8-058d-d6575e6ed1b8++66864f3c-e095-d9c8-058d-d6575e6ed1b8	Ring++66864f3c-e095-d9c8-058d-d6575e6ed1b8++66864f3c-e095-d9c8-058d-d6575e6ed1b8	Valid string.
Send a string lacking ++	Item name not recognized	Ring66864f3c-e095-d9c8-058d-d6575e6ed1b866864f3c-e095-d9c8-058d-d6575e6ed1b8	Ring++66864f3c-e095-d9c8-058d-d6575e6ed1b8++66864f3c-e095-d9c8-058d-d6575e6ed1b8	No such item named Ring66864f3c-e095-d9c8-058d-d6575e6ed1b866864f3c-e095-d9c8-058d-d6575e6ed1b8
Send a string lacking first parameter	Item is not recognized	++66864f3c-e095-d9c8-058d-d6575e6ed1b8++66864f3c-e095-d9c8-058d-d6575e6ed1b8	Ring++66864f3c-e095-d9c8-058d-d6575e6ed1b8++66864f3c-e095-d9c8-058d-d6575e6ed1b8	Invalid string.

3.3. Test Cases for Warehouse HTTP Poller receiving HTTP Response from Web Server

test input	result	input	expected input	output
“Remote Vendor A1” sends purchase request to Web Server. “Warehouse A” is polling.	New purchase detected.	66864f3c-e095-d9c8-058d-d6575e6ed1b8	66864f3c-e095-d9c8-058d-d6575e6ed1b8	John Doe has purchased Ring.
“Remote Vendor A2” has not sent purchase request to Web Server. “Warehouse A” is polling.	No new purchase detected.	66864f3c-e095-d9c8-058d-d6575e6ed1b8	66864f3c-e095-d9c8-058d-d6575e6ed1b8	No new purchases have been made in last 5 seconds.
“Remote Vendor B1” has sent purchase request to Web Server. “Warehouse A” is polling.	No new purchase detected.	66864f3c-e095-d9c8-058d-d6575e6ed1b8	1a59c0ef-2000-dab7-3031-5e6ed1d6cbb8	No new purchases have been made in last 5 seconds.