Object-Oriented Business Process Analysis

Using BPMN to simulate an Object-Oriented Model of a Business Process.

BPM Through Pictures

Introduction

- The purpose of this work is to demonstrate how to create an Object-Oriented implementation of a business process using BPMN.
- The objective is to simulate the execution of a business process using a systems analysis and design model.
- The benefits of a simulation model:
  - Verify the current business process.
  - Show the customer what they are going to receive and get stakeholder approval before building a solution.
  - Demonstrate the solution before development.
  - Discover undesired behavior before building the solution.
  - Understand where the business process may need adjustment.
  - Assess where measurement and control tasks are needed.
  - Provides developers with additional guidance during design and implementation.
  - Provide testers with test cases that can be executed prior to testing.

Overview

- This is a summary of the steps to perform an object-oriented analysis and design using BPMN.
  - Build a current state model of the business, using BPMN.
  - Interact with stakeholders to identify a vision for a future state of the business.
  - Create a class diagram that captures the objects and data to be used by a future state solution.
  - Use a class diagram to create a hierarchy of classes, showing class dependencies and how instances are created and deleted.
  - For each class, create a pool and model the activity of that class using BPMN tasks, events and gateways.
  - Create a sequence diagram showing communication between classes, in particular the creation and deletion of class instances by a parent class.
  - Place all class pools on a BPMN diagram and connect tasks using intermediate events and messages between them.
  - Animate the BPMN diagram to verify (with stakeholders) that the model performs as described in the vision of the future state.

Mapping Between Objects and BPMN

- BPMN does not include the concept of class instances. Use a pool to represent a class.
- Add a workflow to the pool to describe the lifecycle of the class.
- An instance of the class is represented by a token executing within that workflow.
- To create an instance, an external pool or entity, sends an message to the pool.
- This causes a token to start executing within the pool. (Swimlanes may be used to divide the workflow by the objects it interacts with.)
- A subprocess is used to describe the action performed by a task.
- A hierarchy diagram shows which classes create and delete instances of other classes.
- When an instance is created, the start event in its workflow is receives a message. The instance initializes itself by setting its attribute values and creating necessary class instances.
- When an instance receives a deletion message, it executes a delete task and then enters an end event.
- Use the class identifying attribute as the process instance correlation key. This allows you to specify which executing object instance receives the message.

Convert A STD Into A BPMN Workflow

- Start state
- State
- Event
- End state

Tools

- The example is built with Visual Paradigm and Camunda.
- Visual Paradigm is a CASE analysis and design tool, with support for UML (Unified Modeling Language) and BPMN.
- Camunda is a process automation platform with support for BPMN and DMN (Decision Model and Notation).
- Visual Paradigm is used to create the business process diagrams and map them to class diagrams.
- Camunda is used to model class activity and simulate the flow of tokens throughout the solution.
- Visual Paradigm creates the analysis and design. Camunda verifies the analysis and design.

Example Business Process

- The example is of a Automated Menu Ordering System, aimed at improving the efficiency of restaurant operations and reducing the cost of serving customers. (Customer is a group of 1 or more members of the public who will be seated at the same table.)
- The basic steps are:
  - The restaurant opens to the public.
  - A customer enters, and they are greeted by a member of staff (greeter).
  - The greeter takes their name and their needs and shows the customer to a suitable table.
  - When the customer is seated, they are presented with a menu.
  - A server takes their order and assists with general enquiries while they are seated.
  - The server places the customer order with the kitchen.
  - The kitchen prepares the customer ordered food and lets the server know when the food is ready.
  - The server picks up food from the kitchen and delivers it to the customer table.
  - The customer has finished ordering and requests their bill from the server.
  - The server gets the customer bill from the cashier and returns the bill to the customer.
  - The customer hands a payment method to the server.
  - The server takes the payment to the cashier and the cashier validates the payment.
  - The cashier hands a receipt to the server who takes it to the customer.
  - The customer leaves and the server cleans the table to make it ready for a future customer.
  - Closing time arrives and when all customers have left the restaurant, it closes.

Model The Business Process

- The business process includes 3 categories; People, Objects and Activity.
- Use BPMN to model:
  - People with swimlanes.
  - Objects with data stores and data objects.
  - Activity with tasks, events, and gateways.
- Pools group related swimlanes and messages show communication between pools.
- Sequence flows connect related tasks, events and gateways within the same pool.
- Data associations connect objects to tasks.

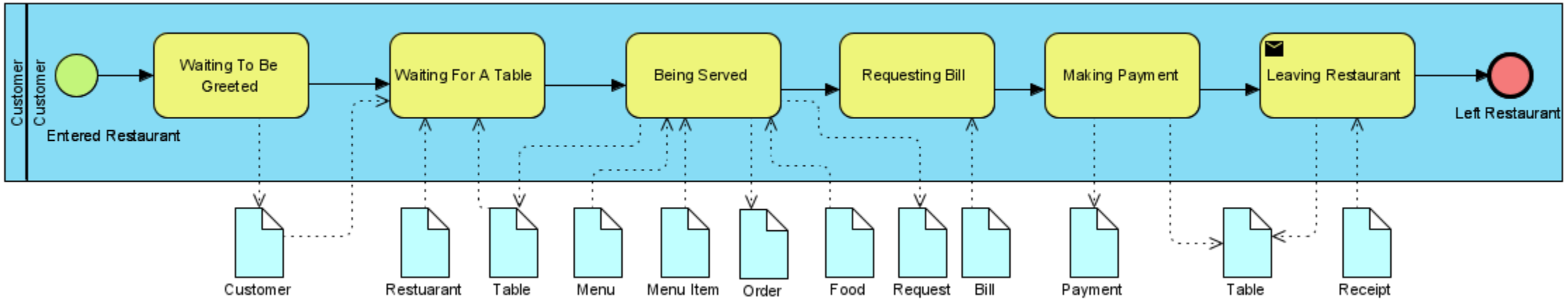
Example BPMN Diagrams

- The following diagrams show the experience of the customer, greeter, server, cashier and chef actors during normal operations. (For this example only normal paths through the business process are modeled.)
- Each actor’s activity is captured within a single swimlane. Each swimlane is embedded within a pool, where a pool represents a customer or a restaurant worker.
- Data objects have been added to show the inputs and outputs of each task in the process.

Customer

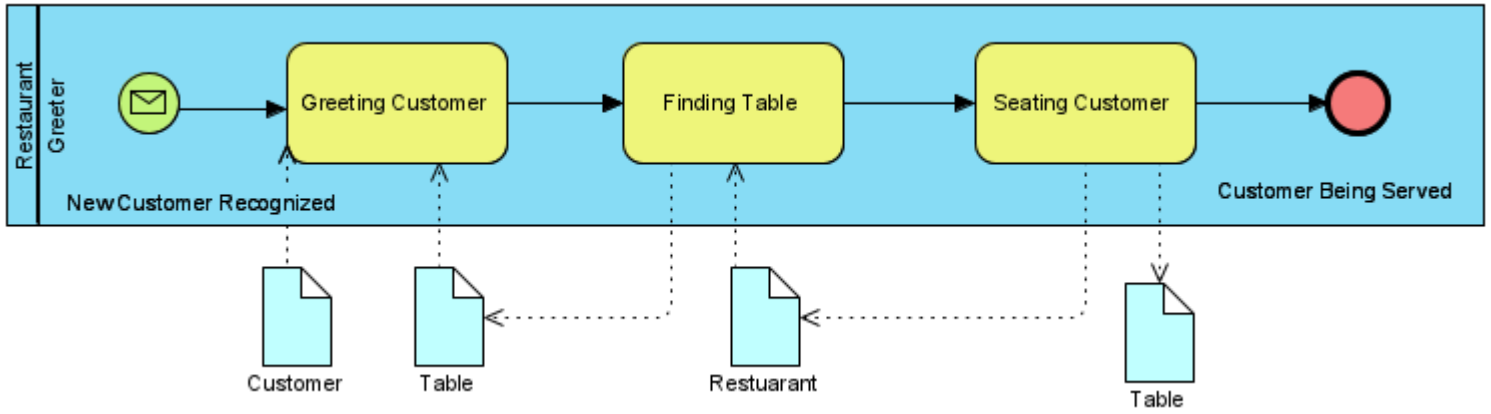
The customer experience is captured with a BPMN diagram.

- A customer enters the restaurant, waits to be greeted, is assigned to a table, is served while at their table, requests their bill, makes a payment and then leaves the restaurant.
- Restaurant, Table, Menu, Menu Item, Order, food, Request, Bill, Payment and Receipt are the objects the customer interacts with while being served.



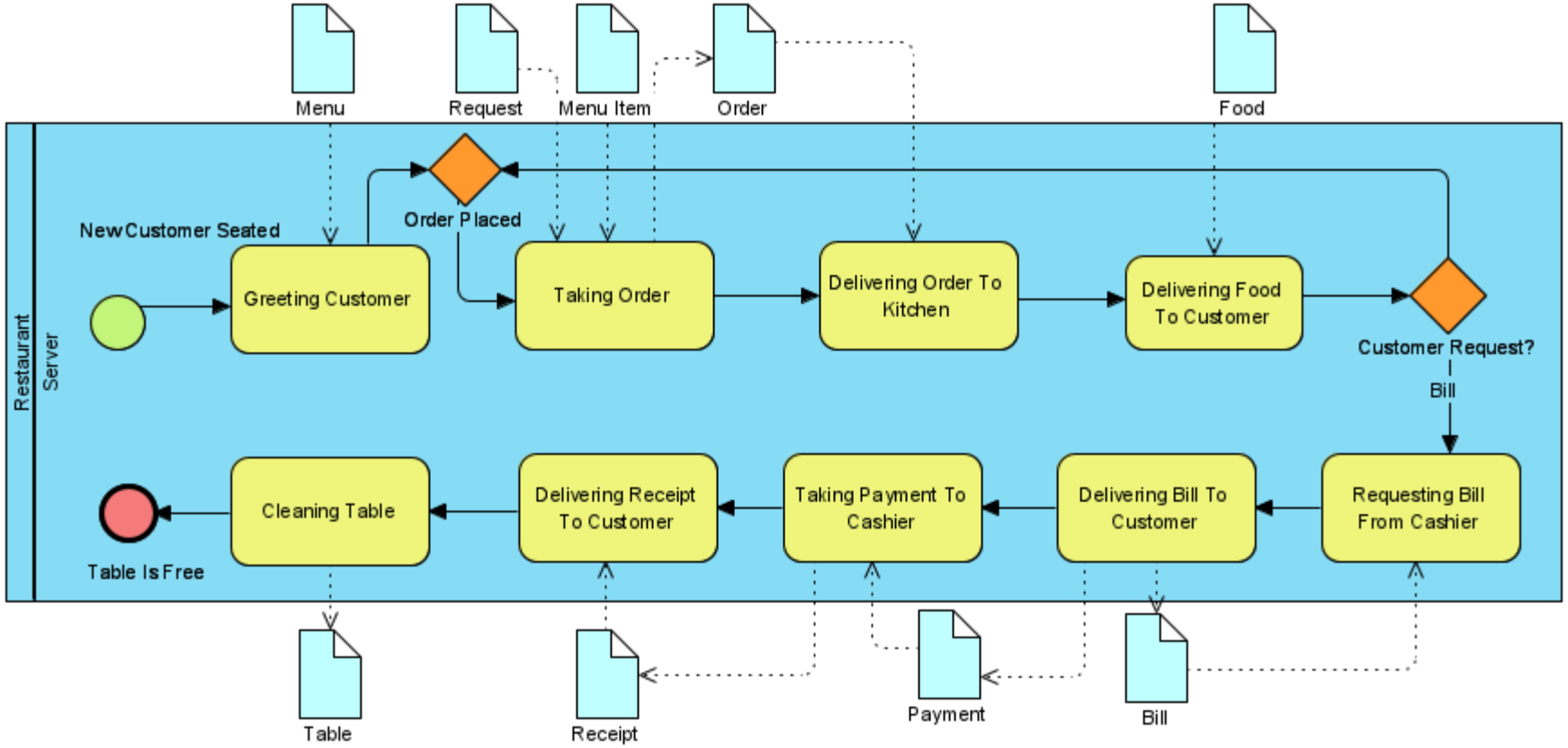
Greeter

- The greeter recognizes a new customer entered the restaurant. The greeter gets the customer needs and finds an appropriate table. The greeter shows the customer to their table.
- Customer, Table and Restaurant are the objects the greeter interacts with while greeting a customer.



Server

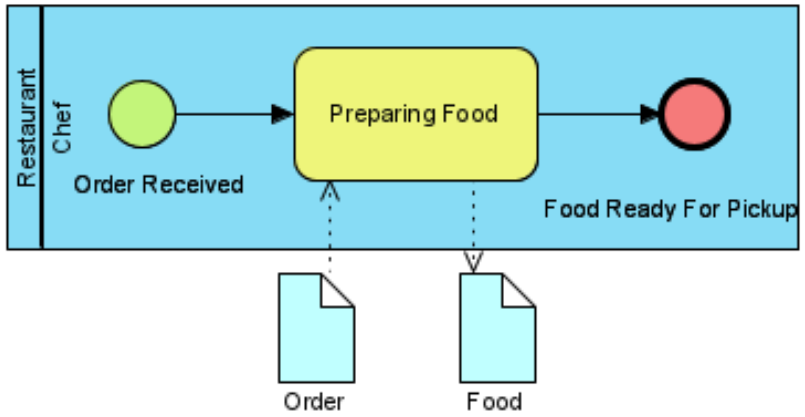
- The server works with the chef and the cashier to feed the customer and take their payment.
- Note that the customer may place several orders.



- Menu, Request, Table, Menu Item, Order, Food, Bill, Payment and Receipt are the objects the server interacts with while service a customer.

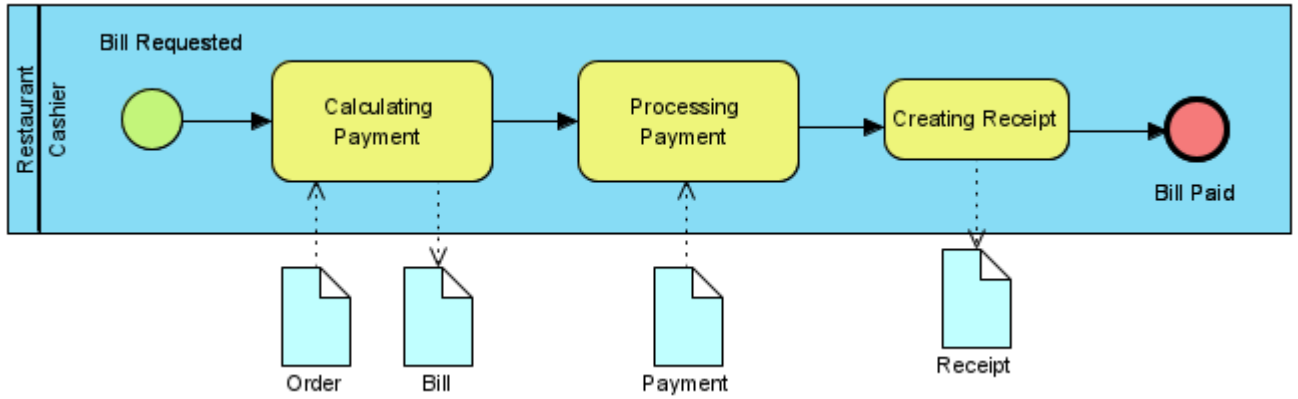
Chef

- The chef receives order for food from the server, prepares the ordered food and informs the server when the food is ready for delivery to the customer.
- The chef interacts with the customer Order and ordered Food.



Cashier

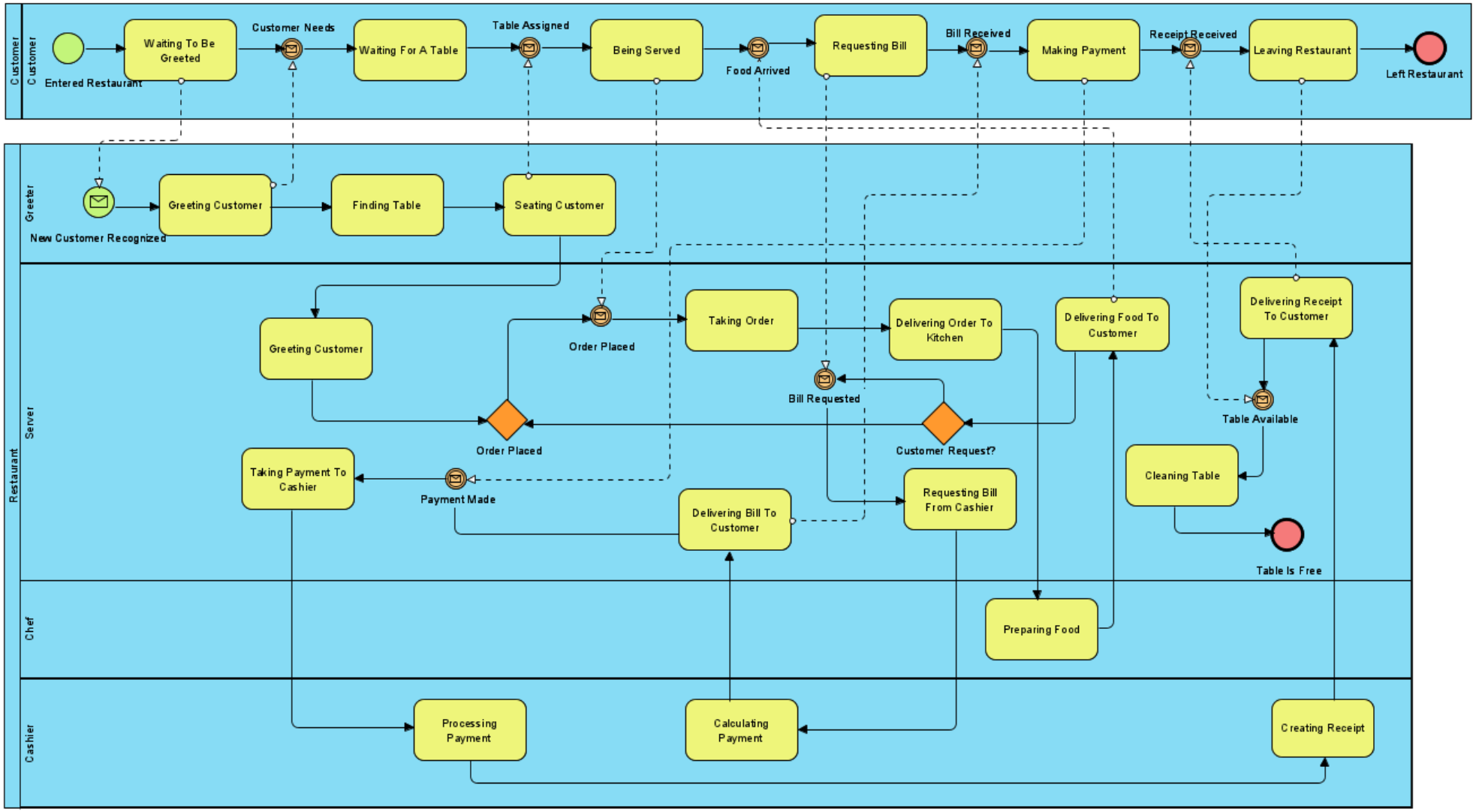
- The cashier receives a request from the server for a customer bill, they calculate the bill, they receive payment for the bill from the server, they process the payment and generate a receipt for the server to return to the customer.
- The cashier interacts with the customer Order, Bill, Payment and Receipt.



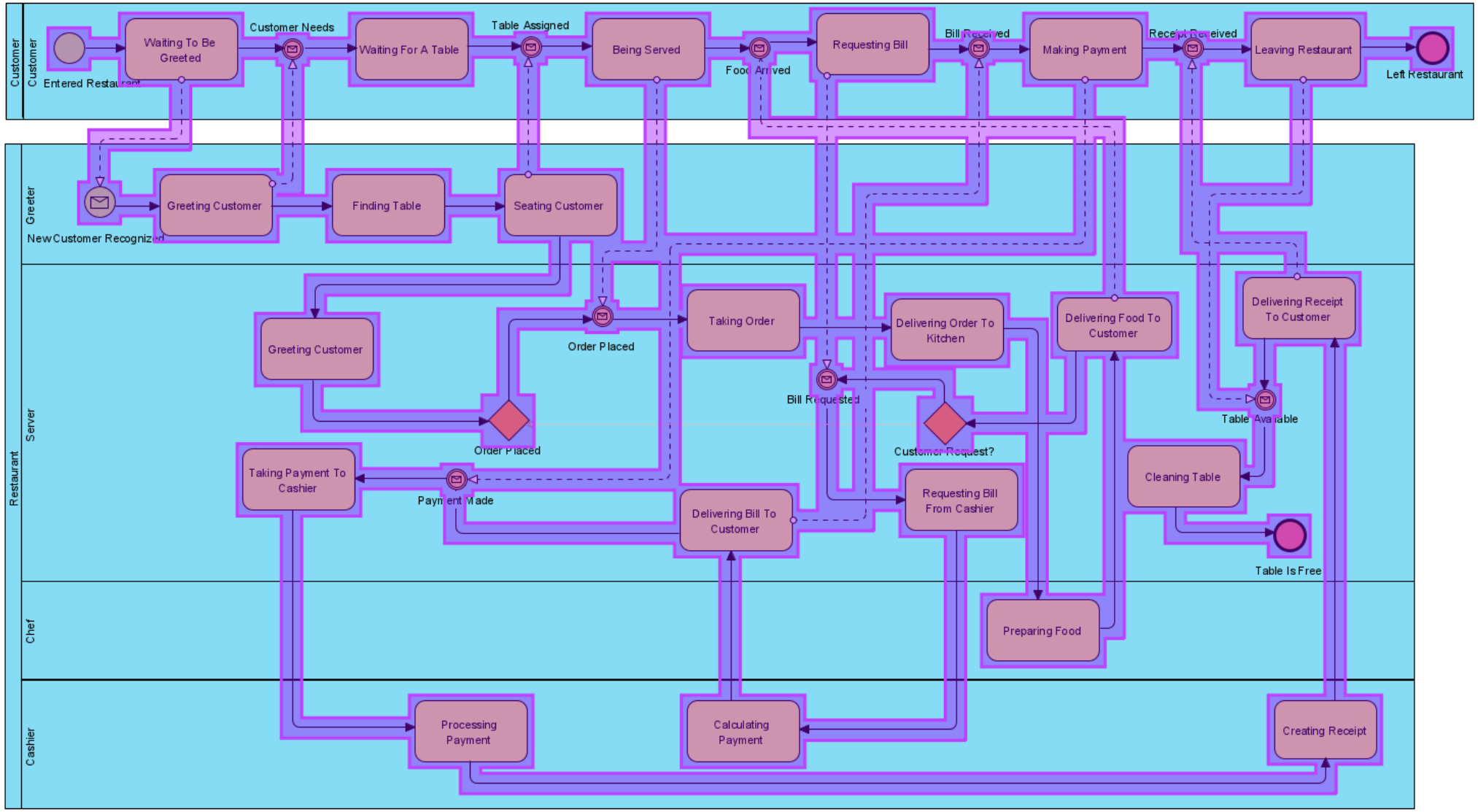
Verify The Business Process

- The actor swimlane diagrams are combined into a single business process diagram.
- Sequence flows are adjusted to show how restaurant workers interact with each other.
- Connections are added to show messages between the customer and restaurant pools.
- The final diagram is animated to verify that all tasks are executed in the correct sequence. (See animation at

Serve Customer Combined Process



Animate The Combined Business Process Diagram



Business Vision

The future state will automate the business process where feasible.

- As a greeter, I want customers to choose their own table so that I don’t need to confirm their table preferences and I spend less time finding their seating requirements.
- As a server, I want customers to easily understand the menu, place their own orders, and pick up their food so that I can reduce my workload and focus on other tasks.
- As a server, I want the customer to be able to request attention, so that I do not have to search for customers needing attention.
- As a chef, I want customers to submit their orders directly to the kitchen so that there is no confusion between what they request and what is prepared.
- As a cashier, I want customers to pay their own bill, so that I can concentrate on other financial operations.
- As a restaurant manager, I want to automatically update table layouts and menu items so that customers only see what is currently available.

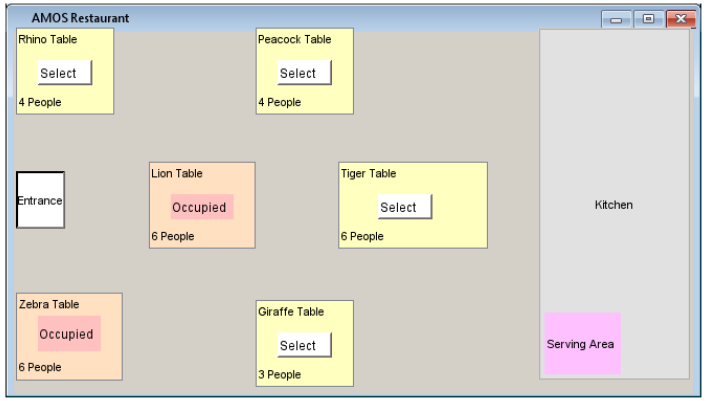
Mockup User Interfaces

Three user interfaces are needed.

- Restaurant Display – The display that is seen by customers when they enter the restaurant. It shows tables in the restaurant, their status and allows the customer to select a suitable and available table.
- Table Display – The display that is seen by a customer while assigned to a table. It requests customer information, displays a menu and an order, allows the customer to copy and move menu items between the menu and the order, allows the customer to place an order, request their bill and make payment. The customer may request assistance from a member of staff.
- Kitchen Display – Ordered menu items are displayed to the staff. Staff are able to update the status of ordered food as it is prepared.

User Interfaces

Restaurant Display



Kitchen Display

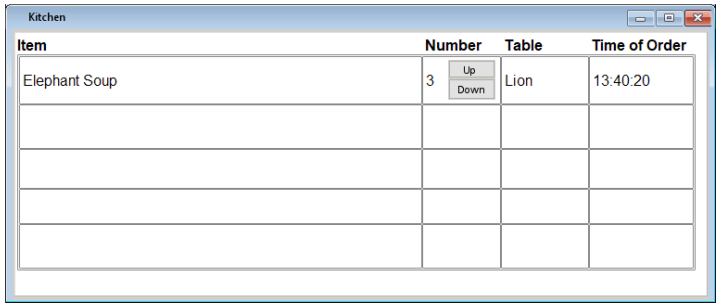
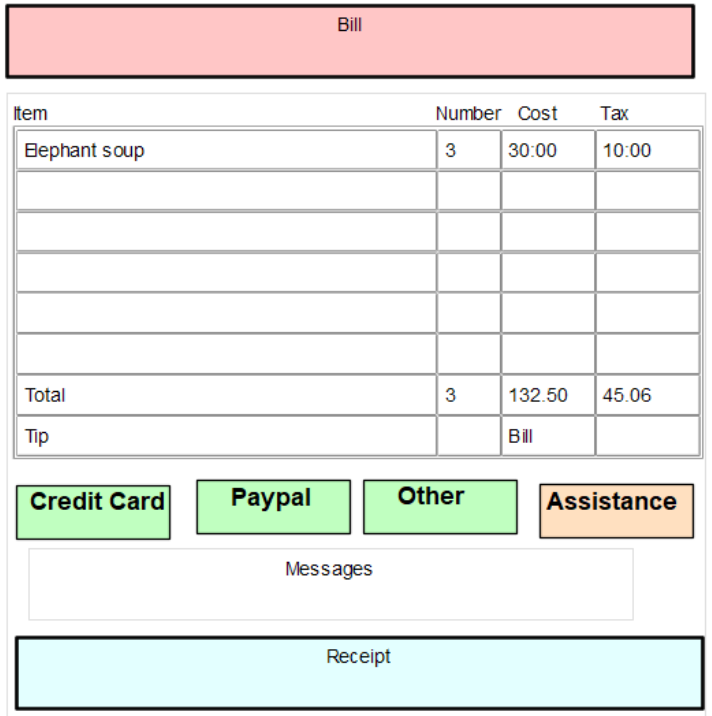
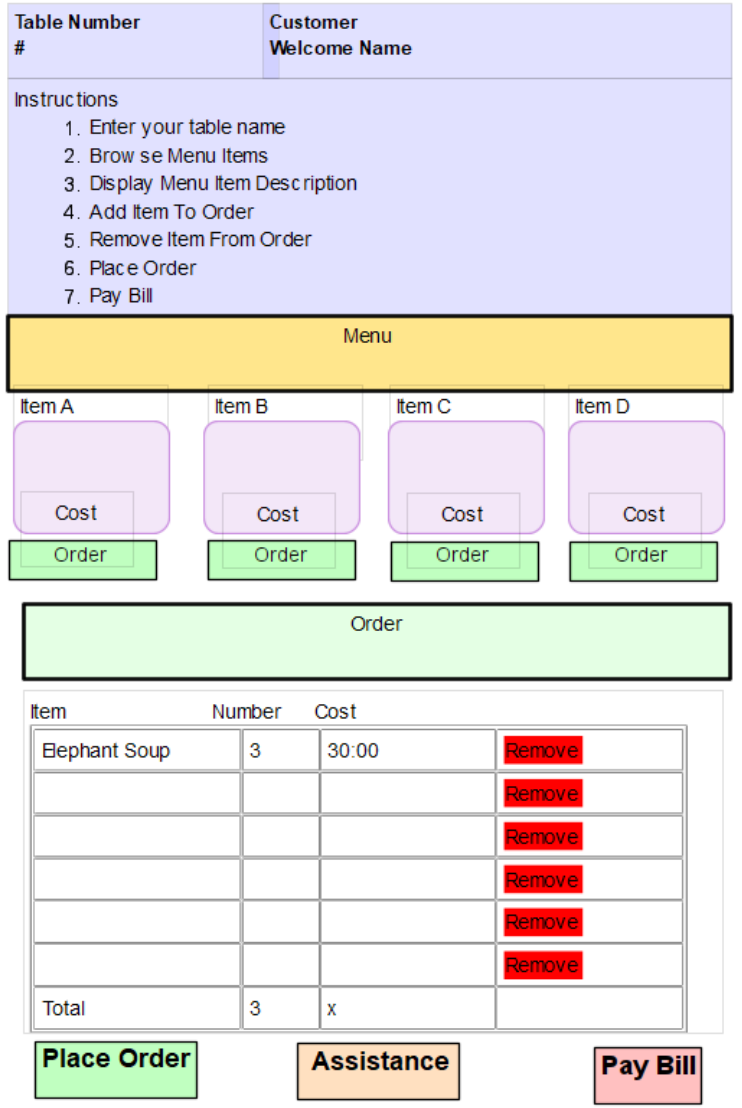


Table Display





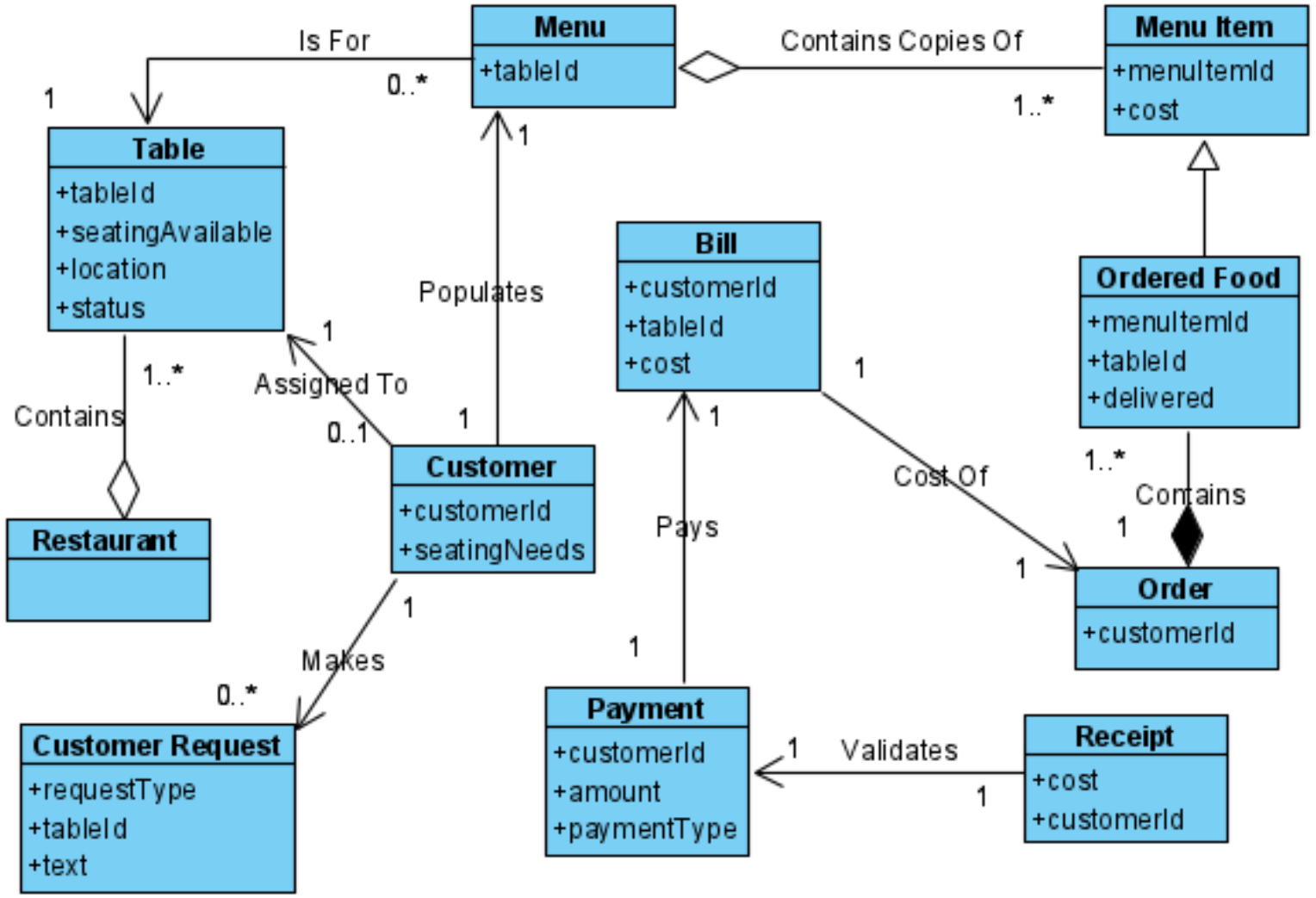
System Model Of The Business Data Objects

With the business process defined, documented and verified:

- Use classes to breakdown the business process functionality and data.
- Document an overview of the process that each class performs in order to satisfy business vision statements.
- For each class in the logical model, create a pool.
- This pool will define the operations performed by the class.
- Add a start and an end event to the pool.
- Add a Create task after the start event and a Delete task before the end event.
- Add tasks between the start and end events, that map out the normal set of steps performed by an instance of that class.

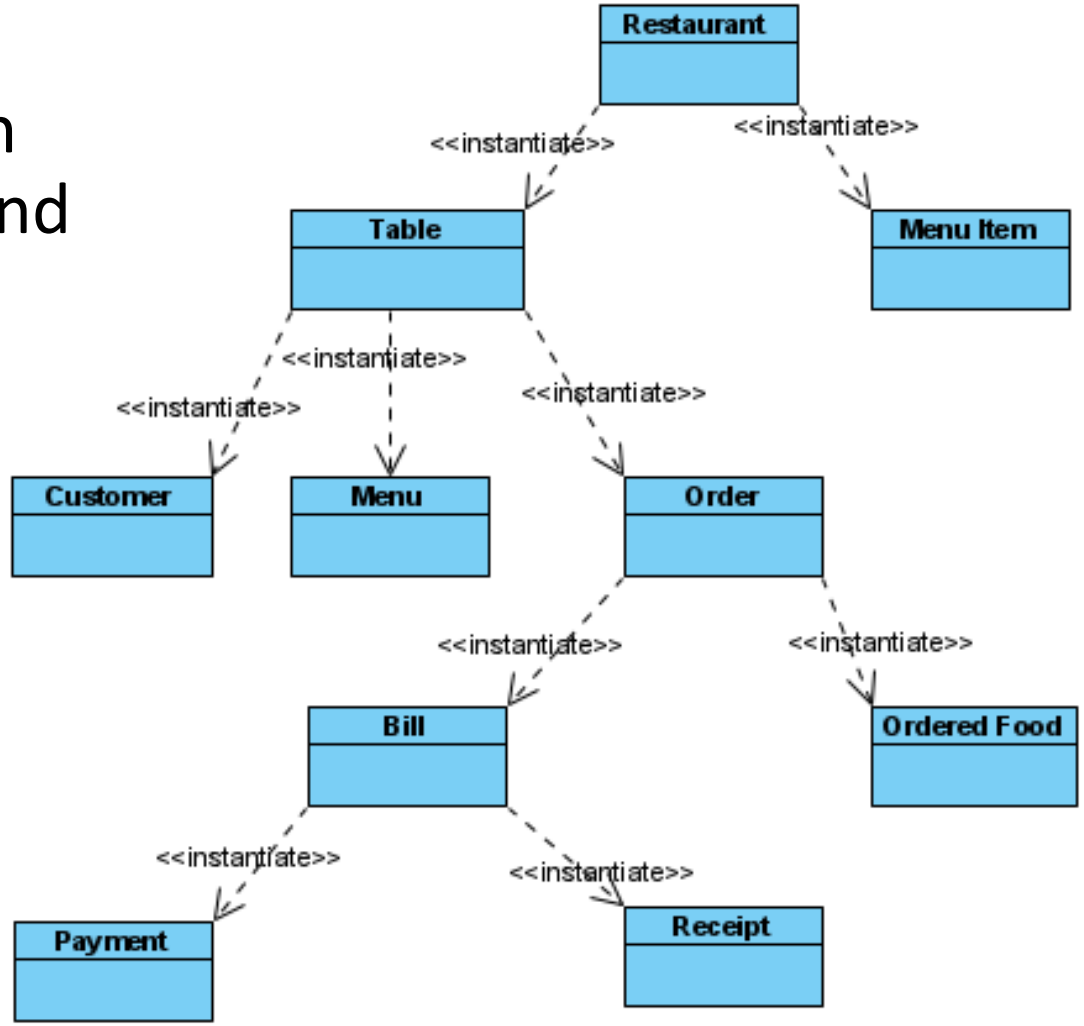
Logical Model Of Business Data Objects

- Place business data objects on a class diagram.
- Add identifying and primary attributes to the classes.
- Show the relationships between classes, and the cardinality of those relationships.



Class Hierarchy Diagram

- The class hierarchy diagram shows class dependencies. An instantiated class is created and deleted by its parent.



Restaurant Activity

The restaurant class is responsible for:

- Maintaining the restaurant display.
- Adding and removing tables.
- Updating the table status.
- Adding and removing menu items.
- Closing the restaurant.

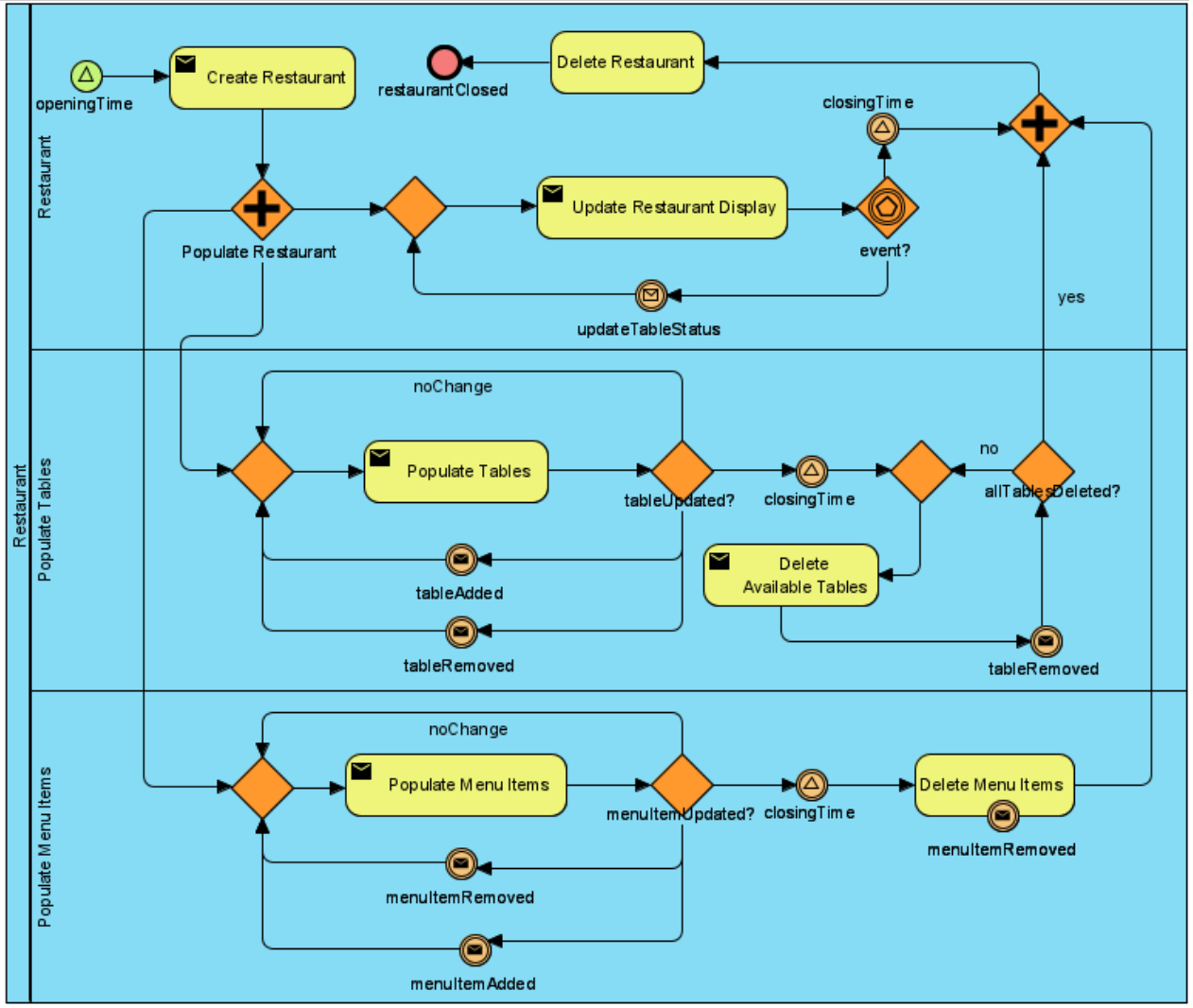
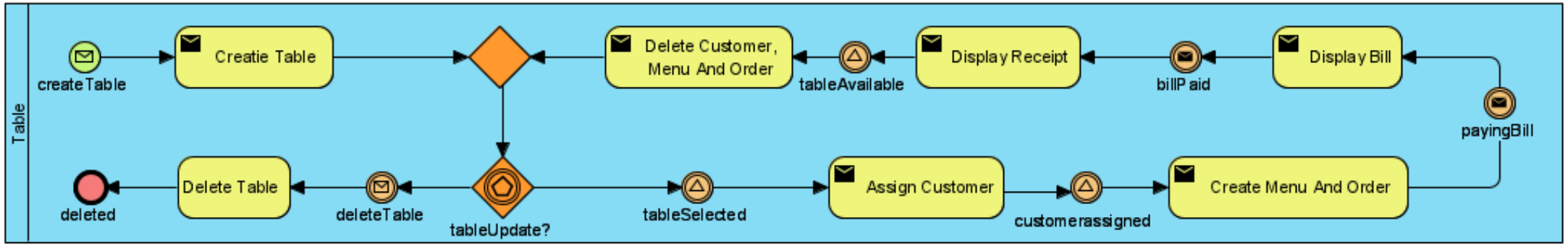


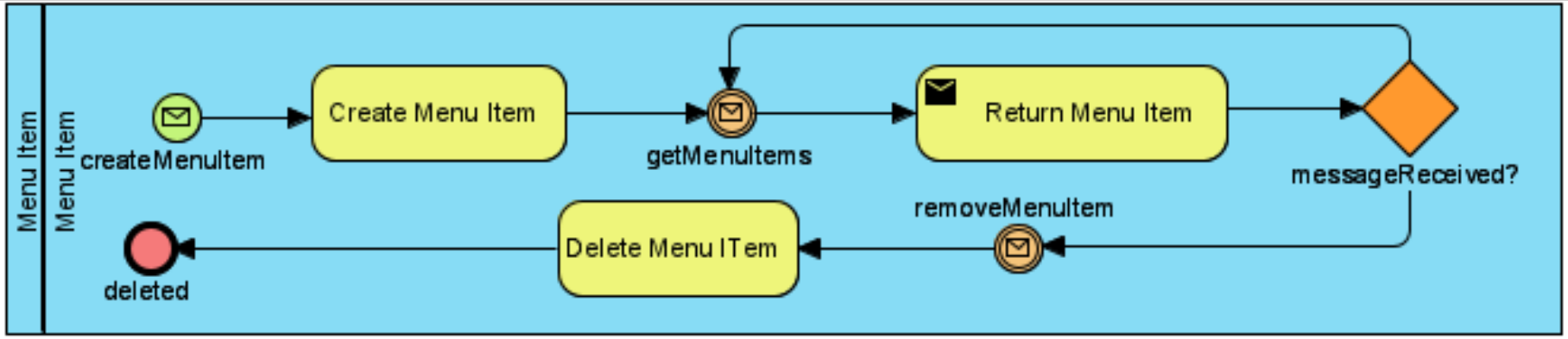
Table Activity



The table class is responsible for maintaining the status of the customer while they are assigned to a table instance.

- The create and delete messages come from the Restaurant.
- Tables can only be deleted if there is no customer assigned.
- The tableSelected, informationEntered, and tableAvailable events come from the user interface.
- The payingBill and billPaid events come from classes within the system.
- A table creates and deletes a customer, a menu, and an order.

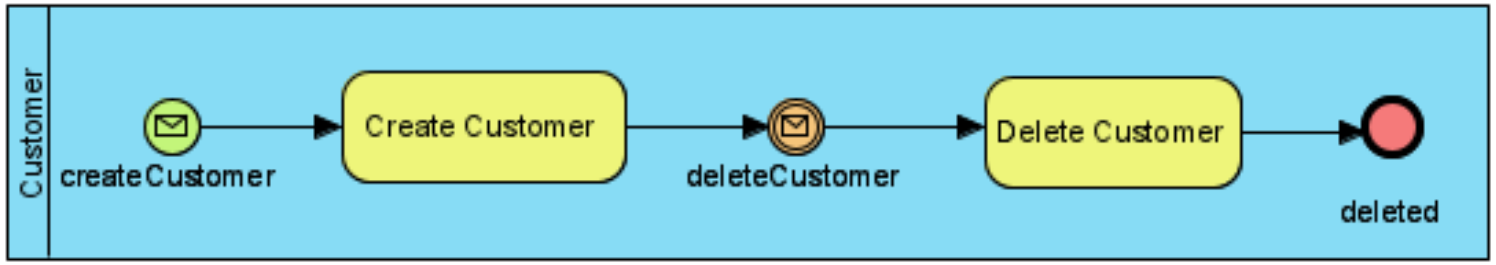
Menu Item Activity



The menu item class is responsible for maintaining menu items.

- The create and delete events come from the Restaurant
- Menu items may be added or removed at any time.
- The getMenuItems event comes from menus before they are displayed to customers.

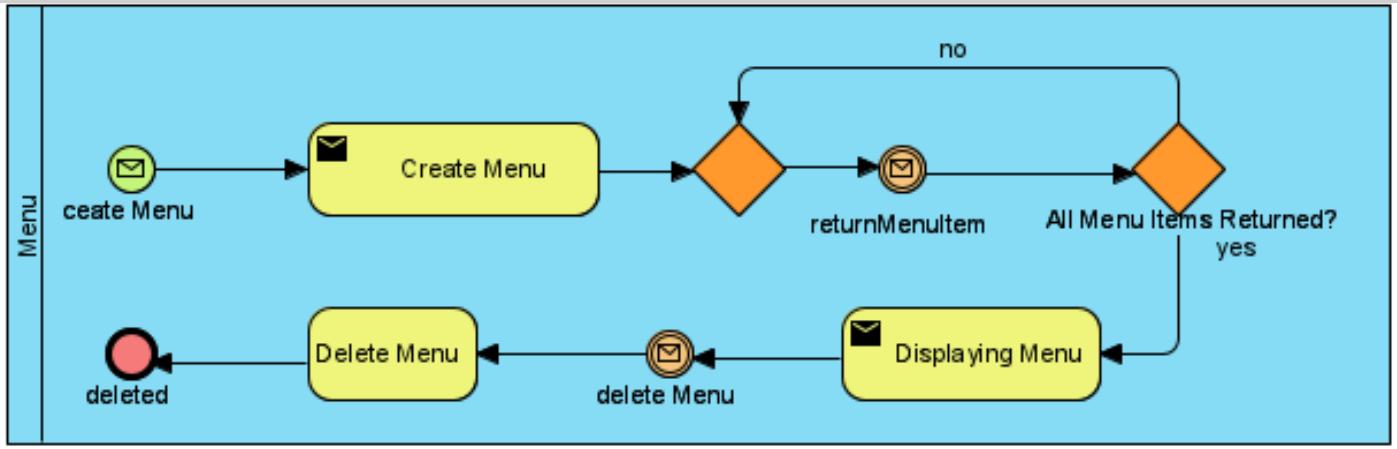
Customer Activity



The customer class collects analytics information.

- The create and delete messages come from the table.
- Customer analytics information is collected while the customer is assigned to a table.

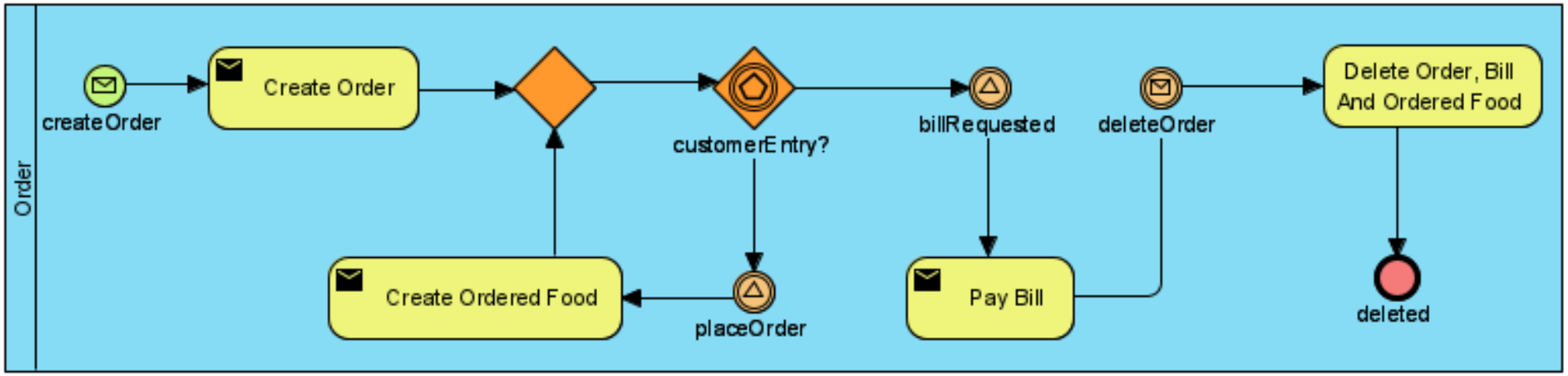
Menu Activity



The menu class contains a list of menu items for display to the customer.

- The create and delete messages come from the table.
- The menu is created when a customer is created and assigned to the table.
- The returnMenuItem event populates the menu with menu items and then displays the menu to the table.
- The menu is deleted when the customer requests their bill.

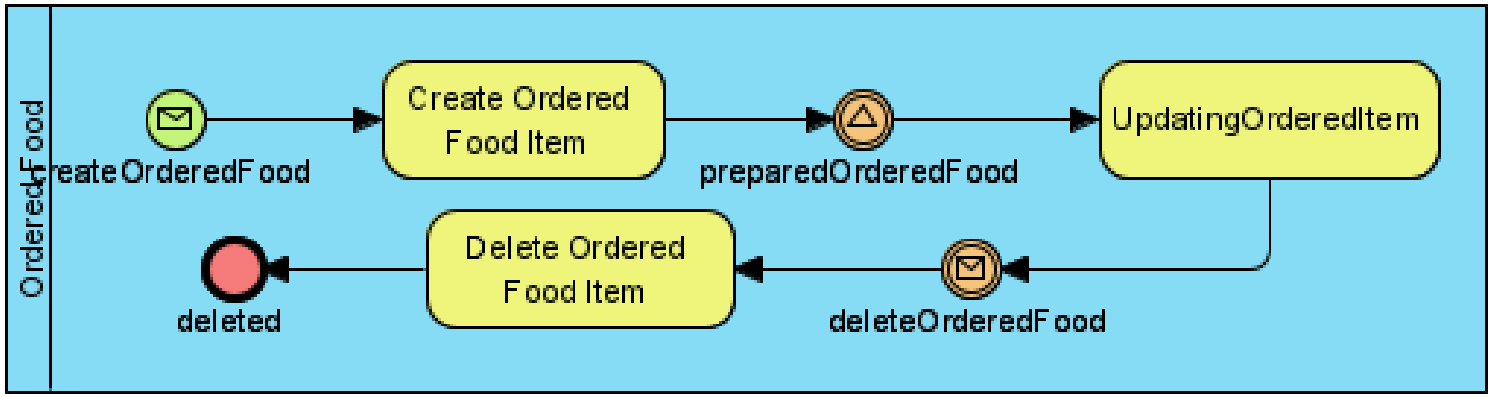
Order Activity



The order class is responsible for ensuring that ordered menu items are prepared and added to the customers bill.

- The create and delete messages come from the Table. When an order is created it is displayed to the customer.
- The placeOrder message comes from the customer and causes ordered menu items to be created.
- The billRequested, event comes from the customer and the order creates an instance of a bill.
- When a bill is paid, order, bill and ordered food items are deleted.

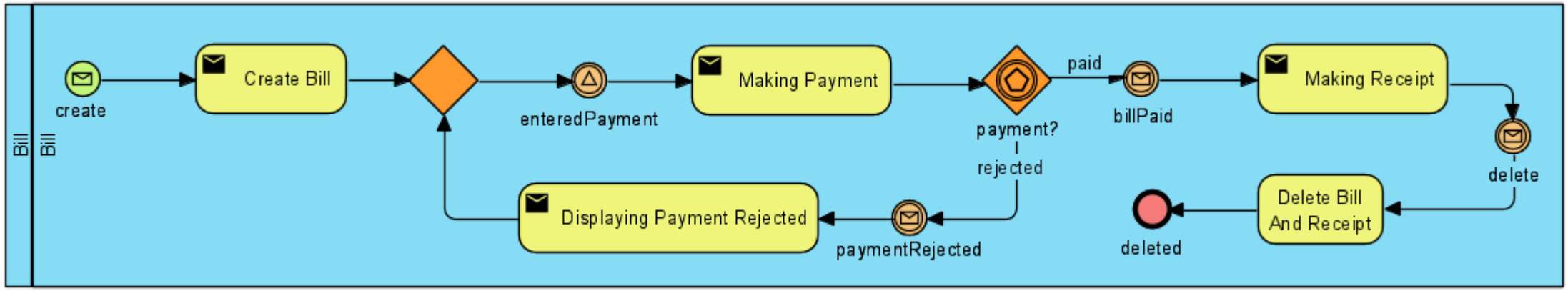
Ordered Food Activity



The ordered food class is responsible for ensuring that menu items on the order are prepared by the kitchen.

- The create and delete messages come from an Order. An ordered food item is created for every menu item on an order.
- The ordered food is displayed to the kitchen.
- The kitchen enters a preparedOrderedFood event as food items are ready for pickup.
- Prepared ordered food is displayed on the customer table.
- Once ordered food has been prepared, it is removed from the list of ordered food on the kitchen display and then deleted.

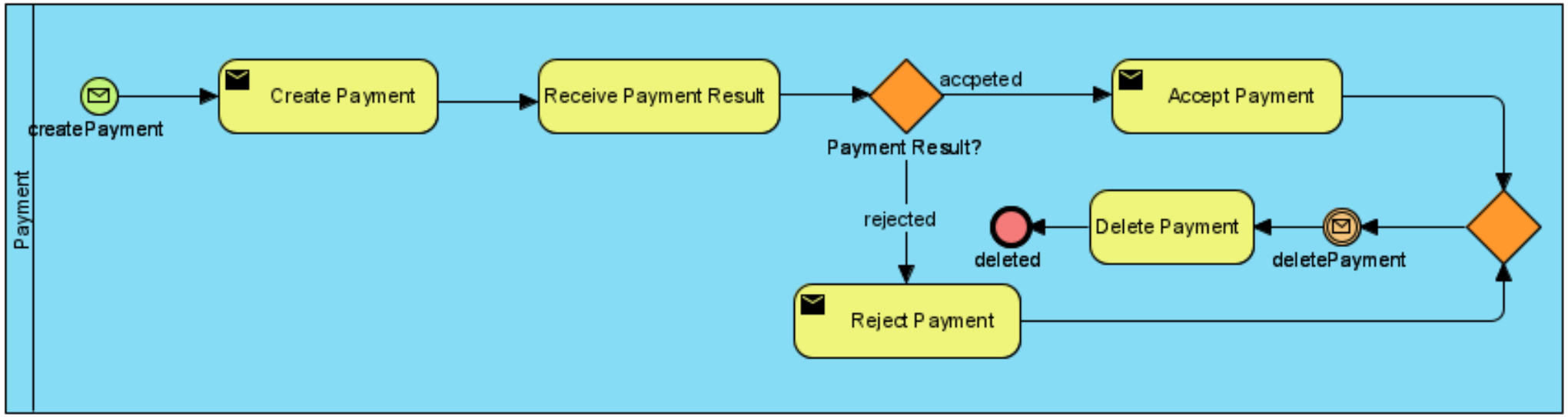
Bill Activity



The bill class displays an itemized list of costs and a total cost to the customer, and requests payment.

- The create and delete messages come from the Order. A bill instance is created when a customer requests their bill. The bill is deleted when payment is accepted. The receipt is deleted along with the bill.
- The bill creates and deletes payments when the customer enters a enteredPayment event.
- The billPaid event comes from a payment instance and causes the payment to be deleted a receipt to be created.
- The paymentRejected comes from a payment instance and causes the payment to be deleted and a new payment request to be displayed.

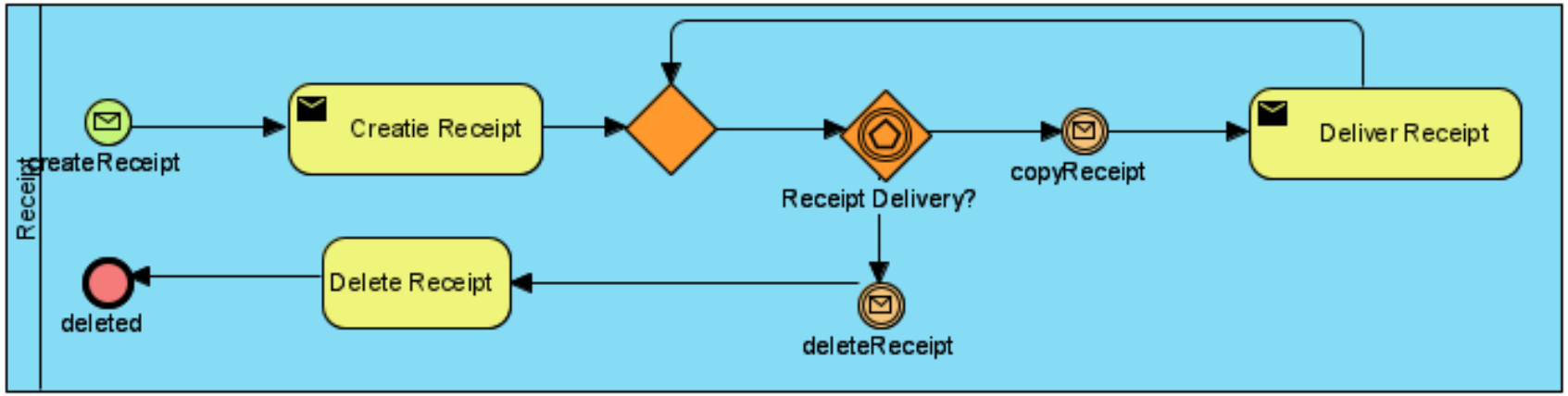
Payment Activity



The payment class takes payment from a customer and sends it to a payment processing system for validation.

- The create and delete messages come from the Bill. A payment instance is created everytime a customer enters a payment method into the system.
- The payment is validated by an external payment processing system, which returns the validation result and a reason for rejection.
- The result is displayed to the customer, at which point the payment instance is deleted.

Receipt Activity

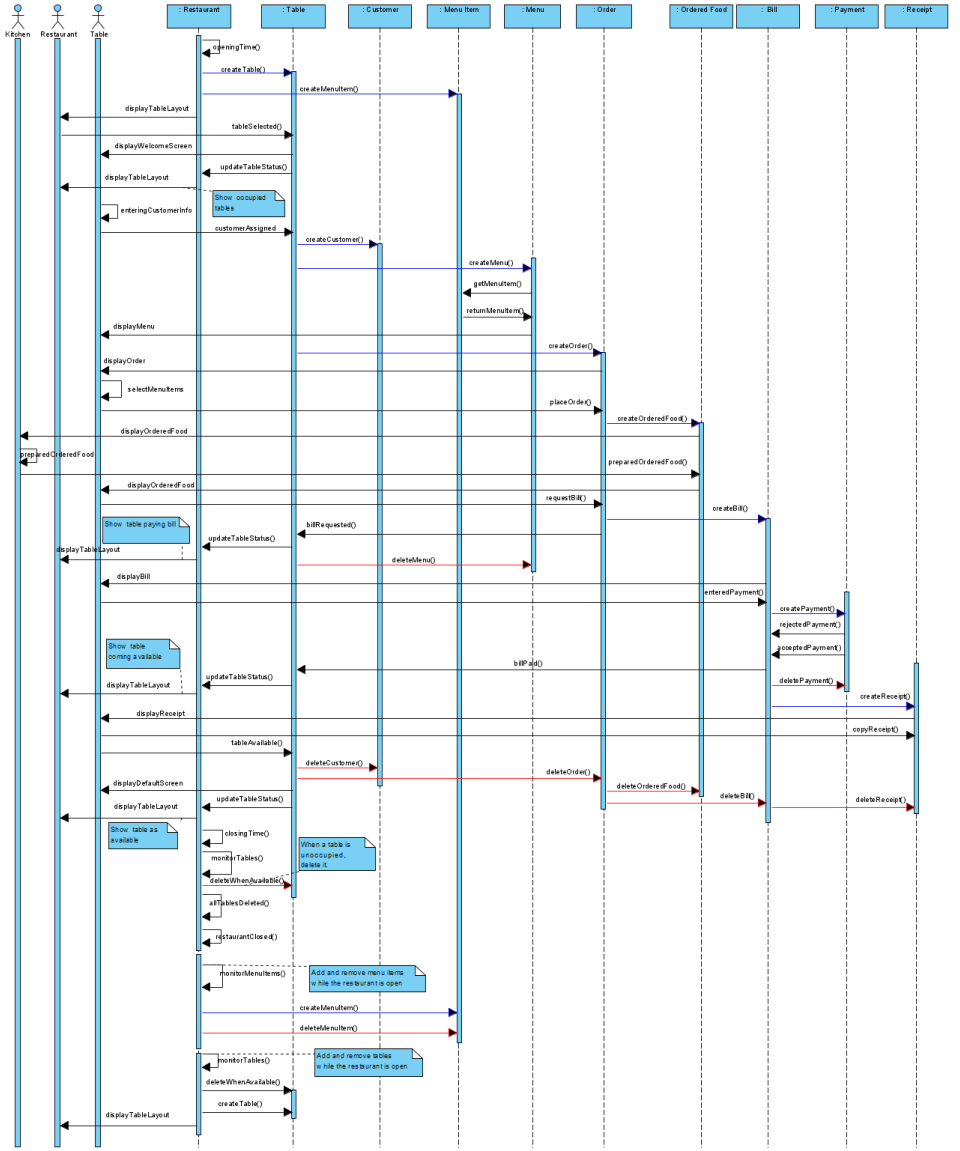


The receipt class creates, displays and copies a receipt for the customer payment.

- The create and delete messages come from the Bill. A receipt is created when a payment is accepted and deleted when the customer leaves the table.
- The receipt may be delivered to the customer using various methods, upon request.

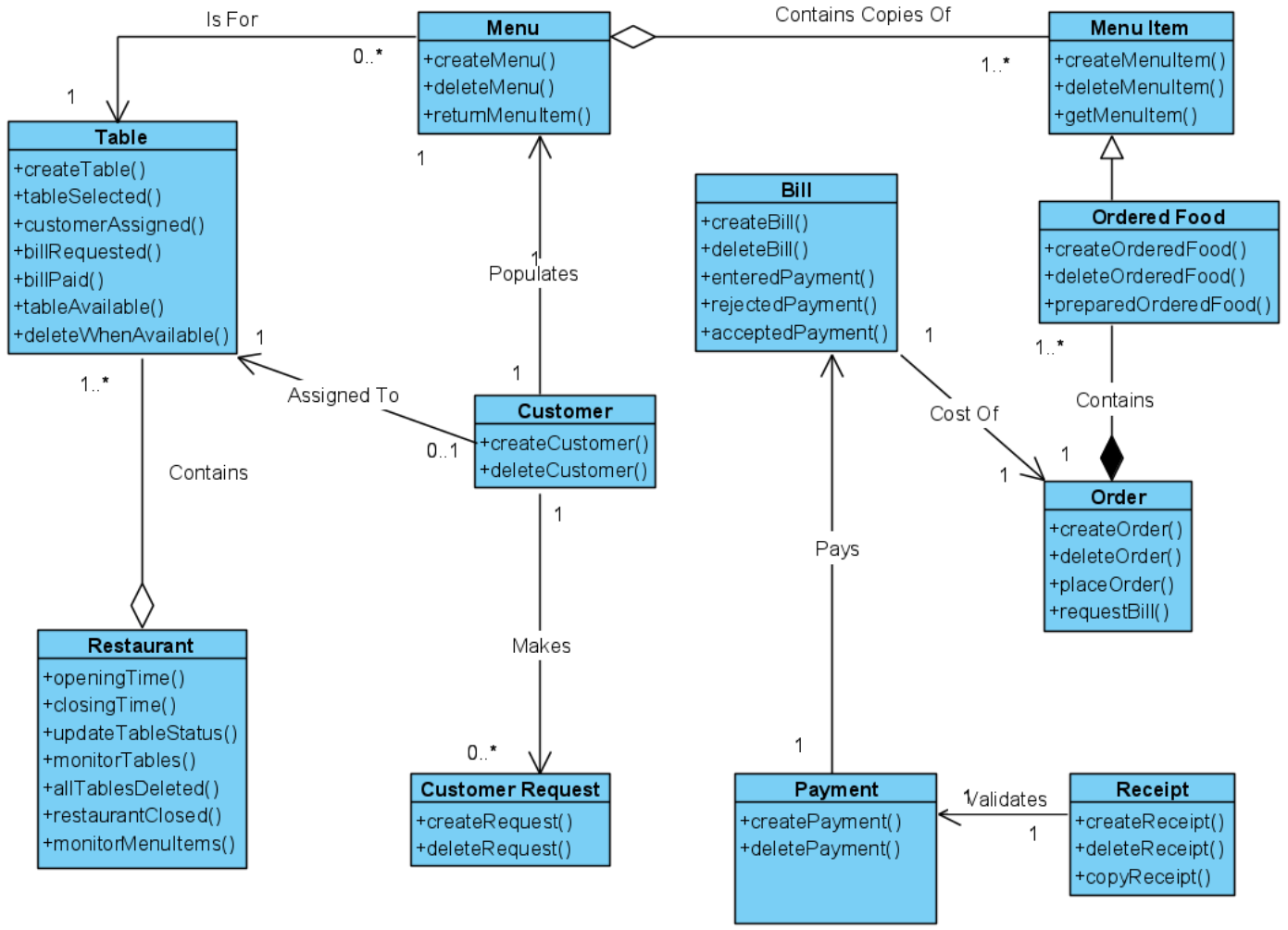
Class Communication Diagram

- The class communication diagram shows a sequence of messages passed between class instances.
- User interfaces have been added to show events displayed to a user of the system.
- External Payment Processing, Delivery and Restaurant Configuration systems are not shown.
- The message names match the event names shown in the BPMN process diagrams.



Add Operations To Classes

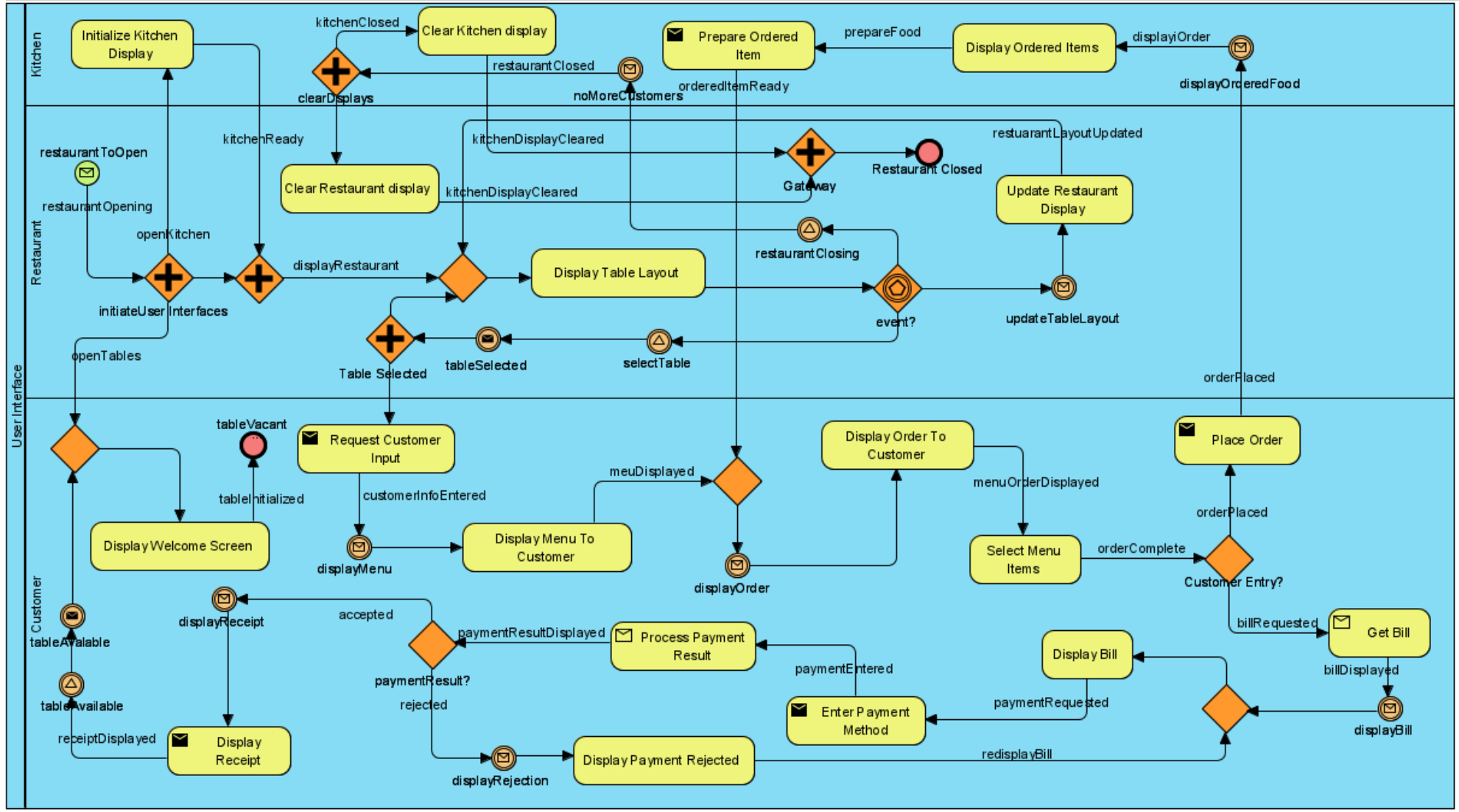
- For each class instance timeline on the communication diagram:
- Extract all messages entering the timeline.
- Create an operation for the message in the class representing the timeline instance.



User Interface Activity

- Create a process diagram showing the steps performed by the 3 user interfaces.
- Create a BPMN diagram.
- Add a pool to represent the user interfaces of the system.
- Add a swimlane for each user interface.
- Add a start event and end event to the restaurant user interface.
- Using the communication, add intermediate events to represent each message received by the user interfaces.
- Connect events with tasks and decisions to show the complete process for serving customers, from when the restaurant opens to when it closes.

User Interface Activity



List Of Process Diagram Animations

- Restaurant animation
- Table animation
- Menu Item animation
- Menu animation
- Order animation
- Ordered Food animation
- Bill animation
- Payment animation
- Receipt animation
- User Interface animation.

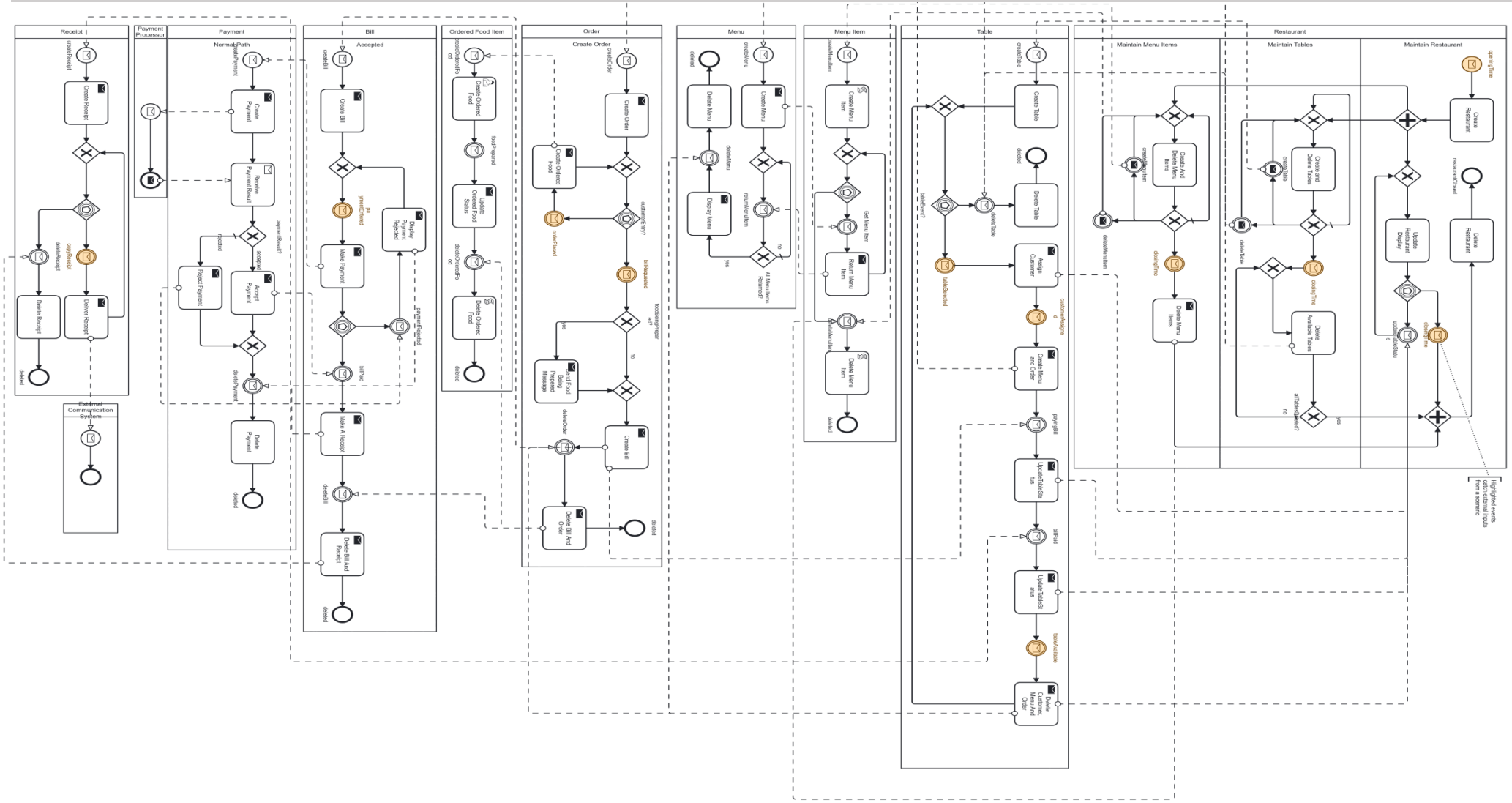
Simulate The system Activity

- *Export the class activity diagrams from Visual Paradigm.
- Import the class activity diagrams into Camunda.
- Combine all class activity diagrams into a single Camunda BPMN diagram.
- Use message flows between pools, to connect class communication.

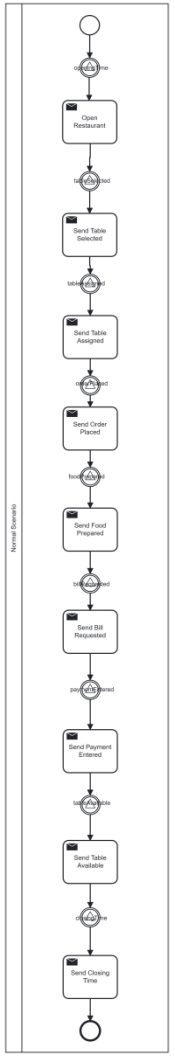
- Create a scenario containing a sequence of user interface inputs.
- Connect the user interface scenario to the class activity using message flows.
- Execute the scenario.

(*Customer and Customer Request classes are not included.)

Combined Class Activity Diagram, With Messages



Create Scenarios To Simulate



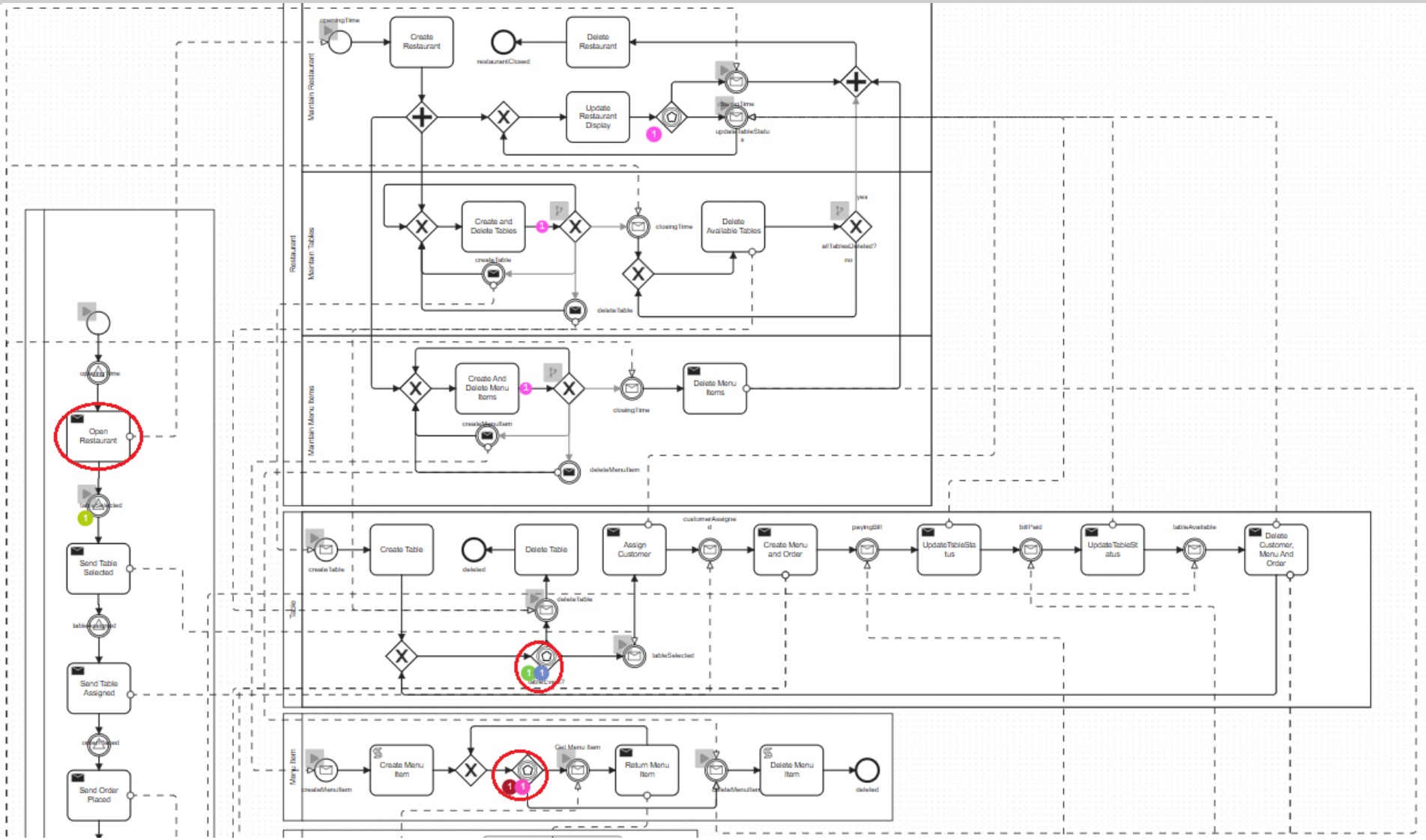
- A scenario shows a sequence of inputs to the systems. Each task in the scenario is connected to a catching message event in the process.
- The basic scenario shows:
  - The restaurant opens.
  - A table is selected by a customer.
  - The customer is assigned to the table.
  - The customer selects menu items and places an order.
  - The ordered food has been prepared by the kitchen.
  - The customer requests their bill.
  - The customer enters a payment method.
  - The server makes the table available for the next customer.
  - The restaurant closes.

Execute The Simulation

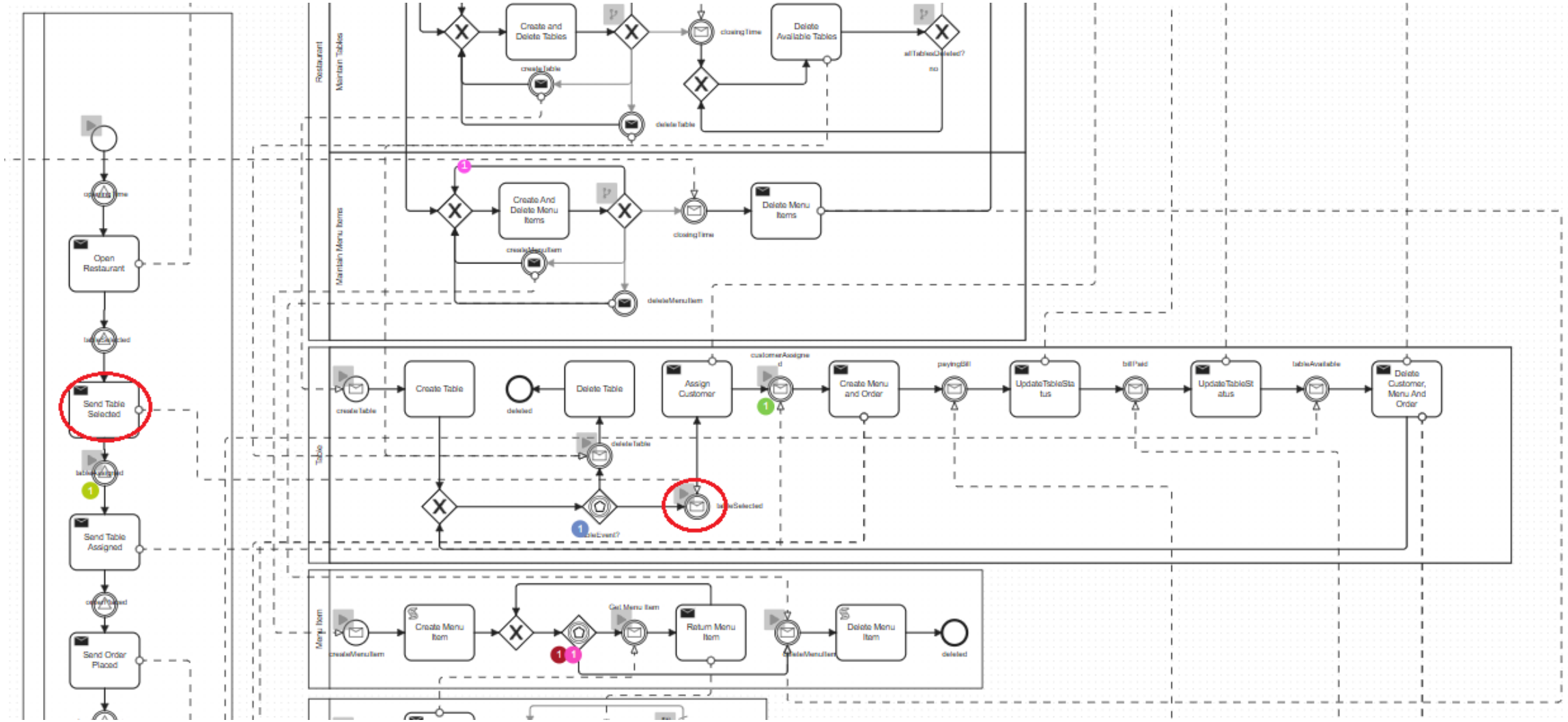
Simulation is performed using the BPMN-JS token simulation plugin for Camunda. The steps to execute the basic scenario.

- Combine the system process and user interface scenario on a single diagram.
- Connect tasks in the scenario to the appropriate intermediate events in the process, using message flows.
- Start the simulation and generate an openingTime event.
- Create 2 Tables and 2 Menu Items.
- Execute tableSelected to reserve a table.
- Execute tableAssigned to create a Menu and an Order.
- Ensure all menu items have been returned and execute orderPlaced to create Ordered Food.
- Execute foodPrepared to update Ordered Food.
- Execute billRequested to create a Bill and prevent more food from being ordered.
- Execute paymentEntered to create a Payment, process the Payment, and create a Receipt.
- Execute tableAvailable to delete the Order, Menu, Bill and Receipt, and make the Table available.
- Execute closingTime to delete Tables, Menu Items and the Restaurant.

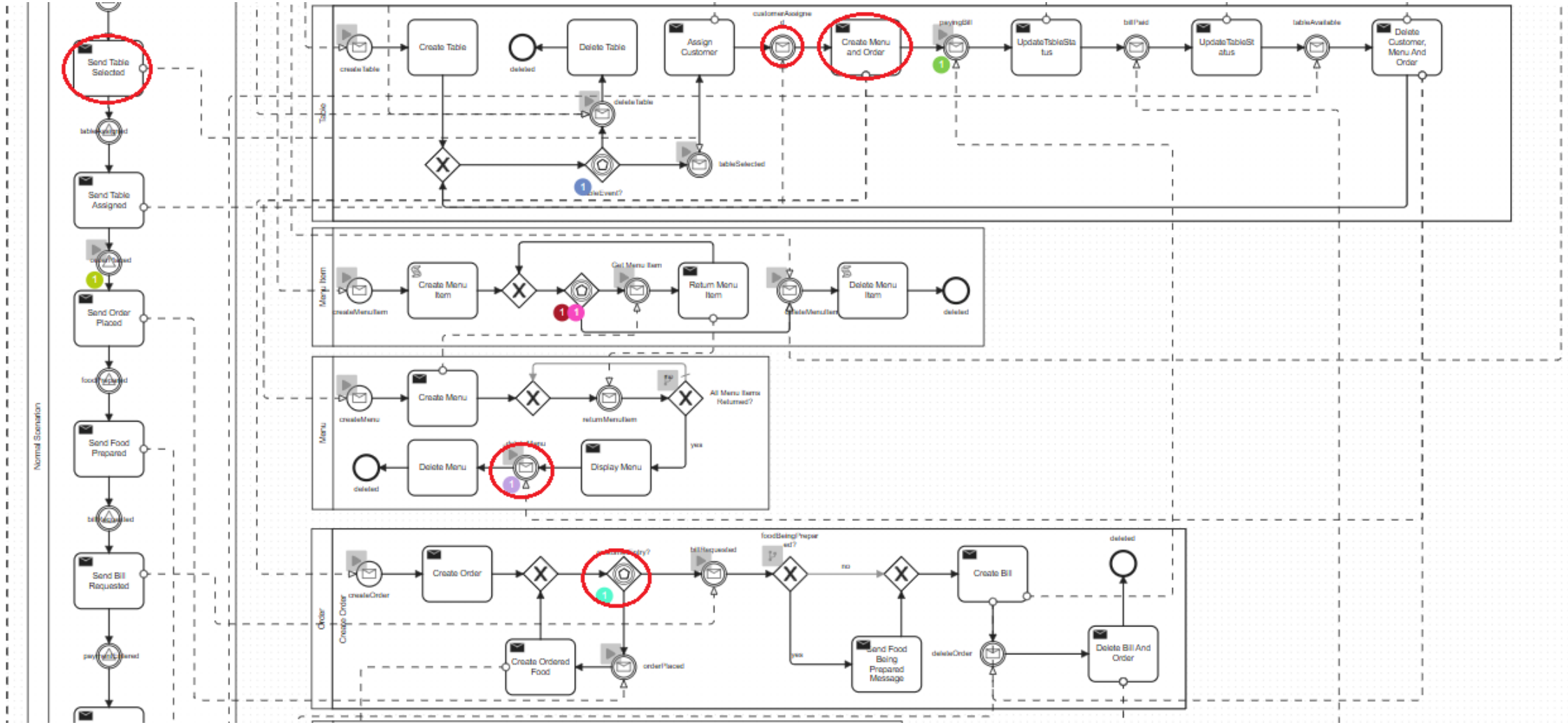
Create 2 Tables And 2 Menu Items



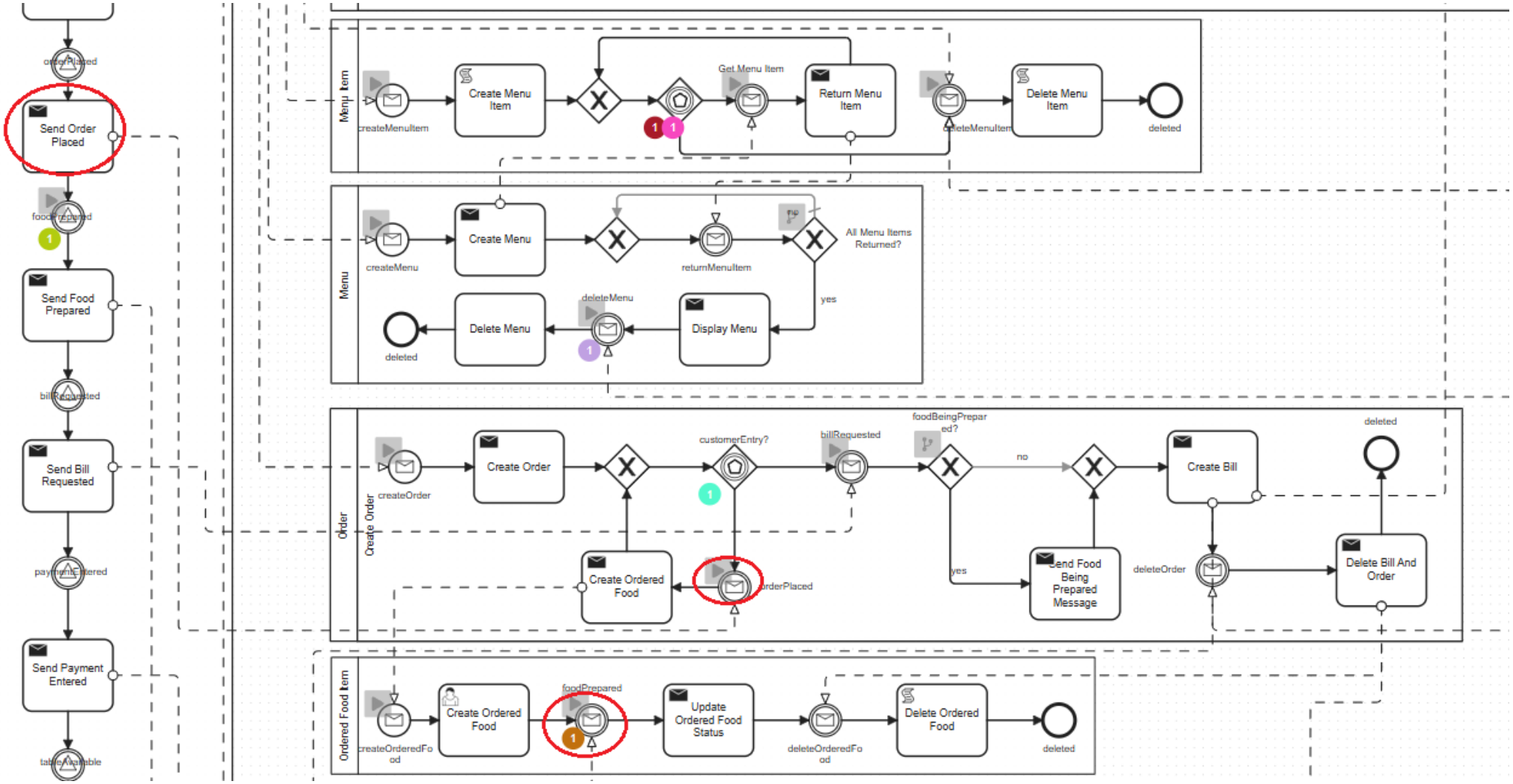
Customer Selects A Table



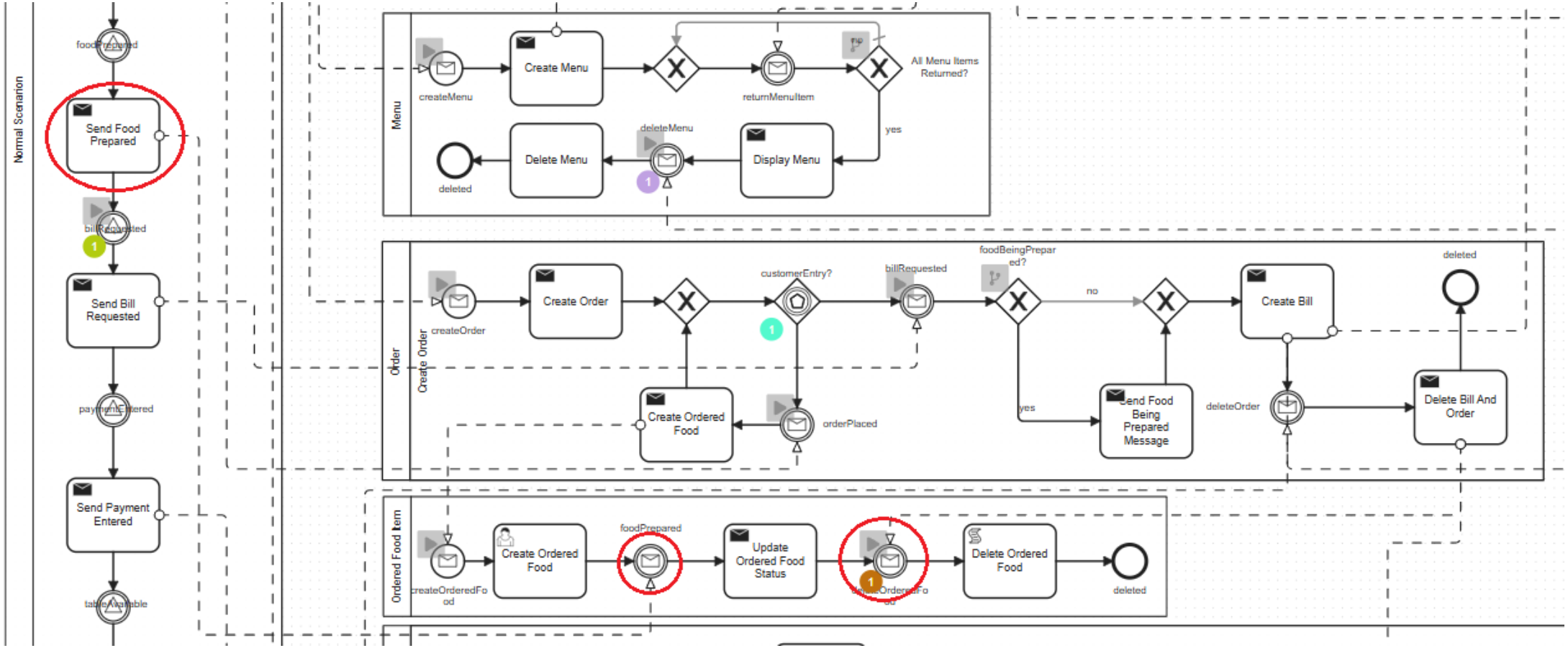
Customer Enters Their Information



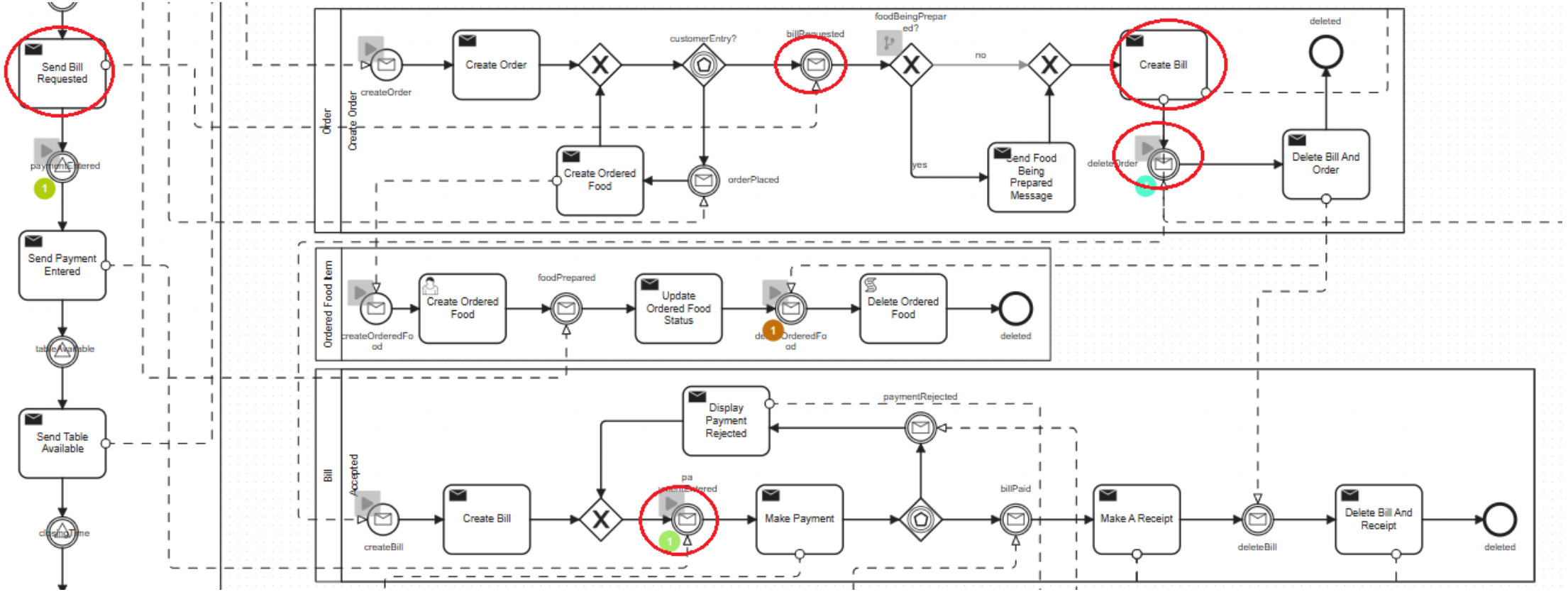
Customer Places Their Order



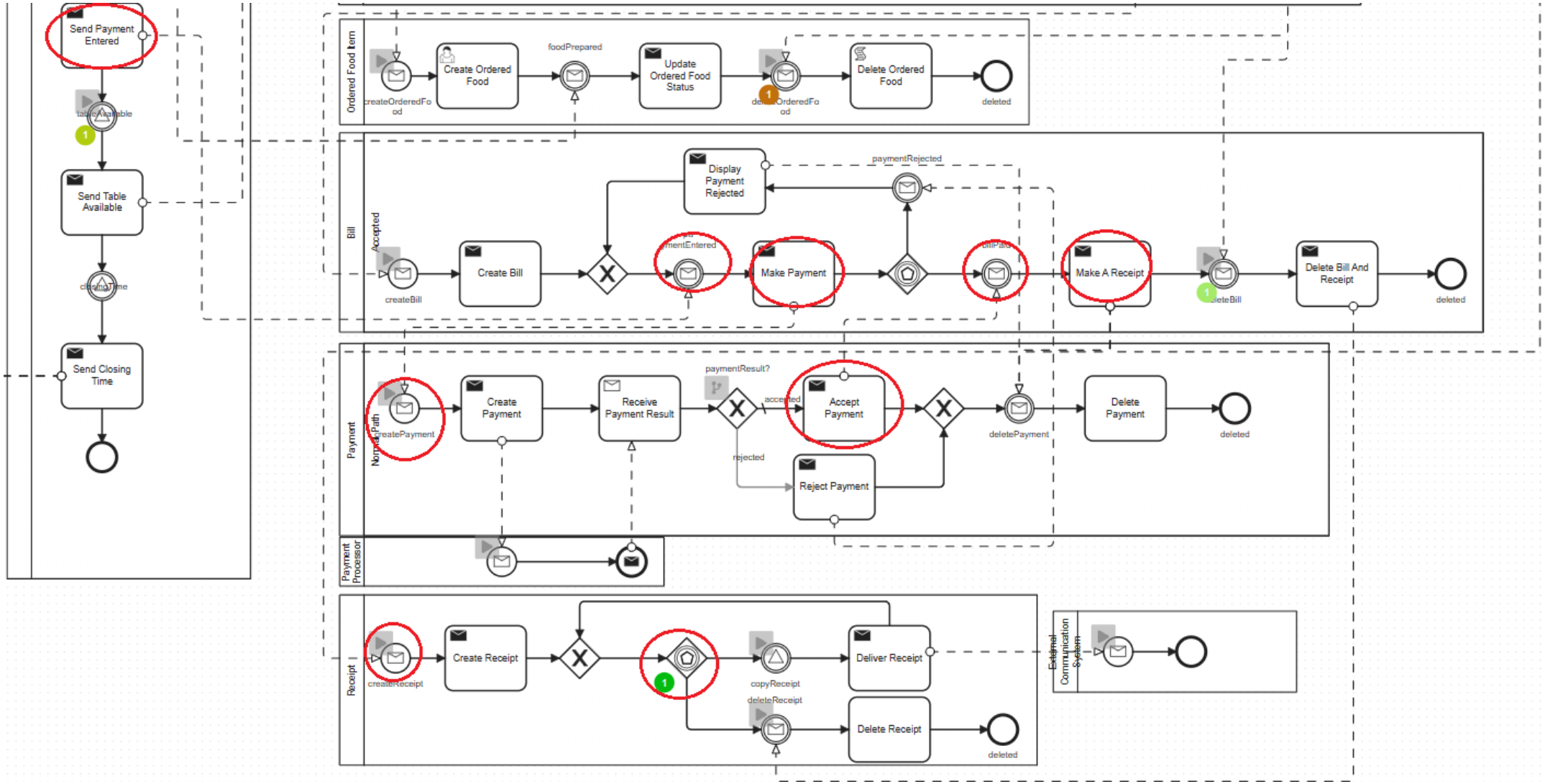
Kitchen Prepares the Ordered Food



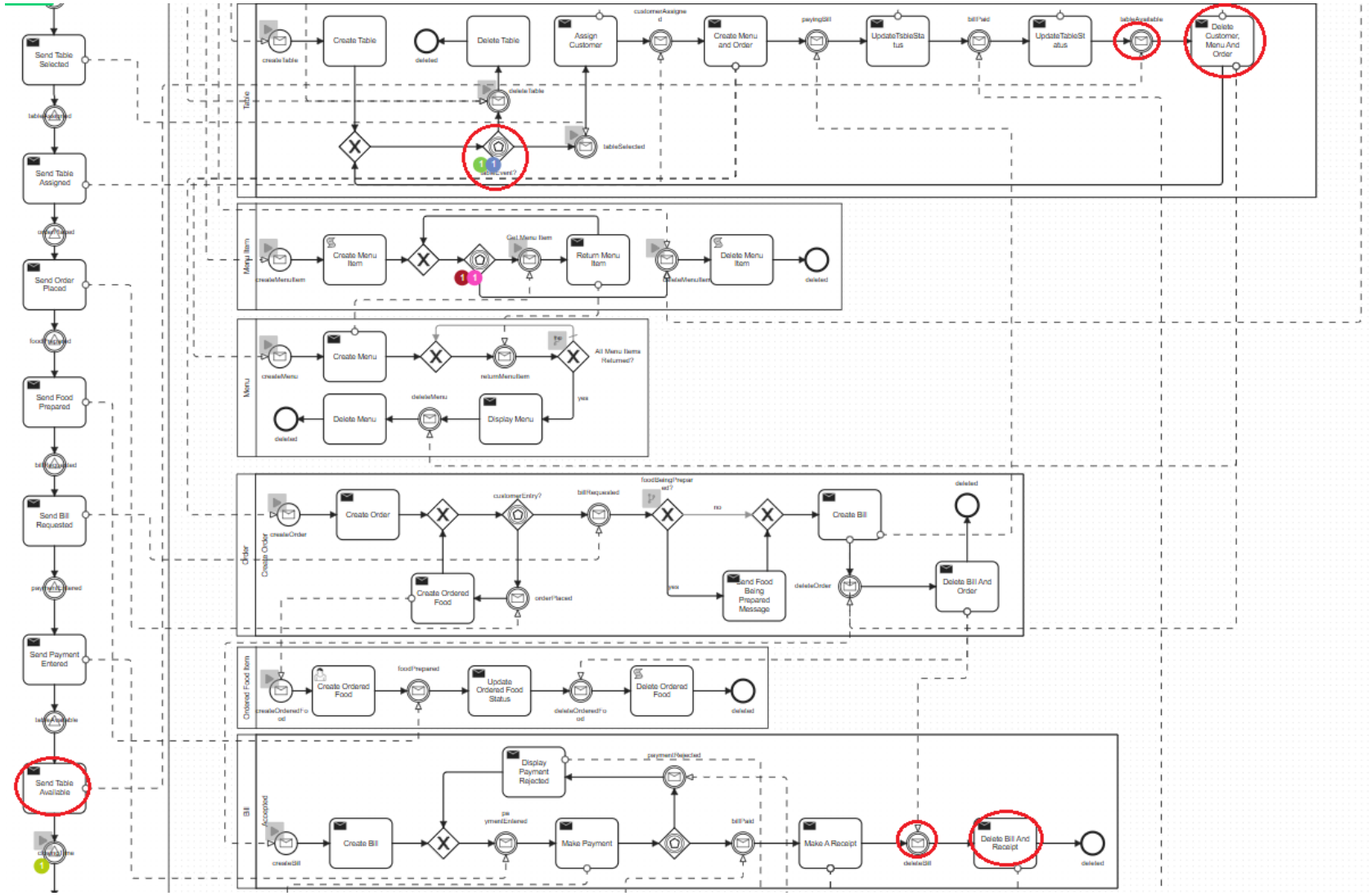
Customer Requests Their Bill



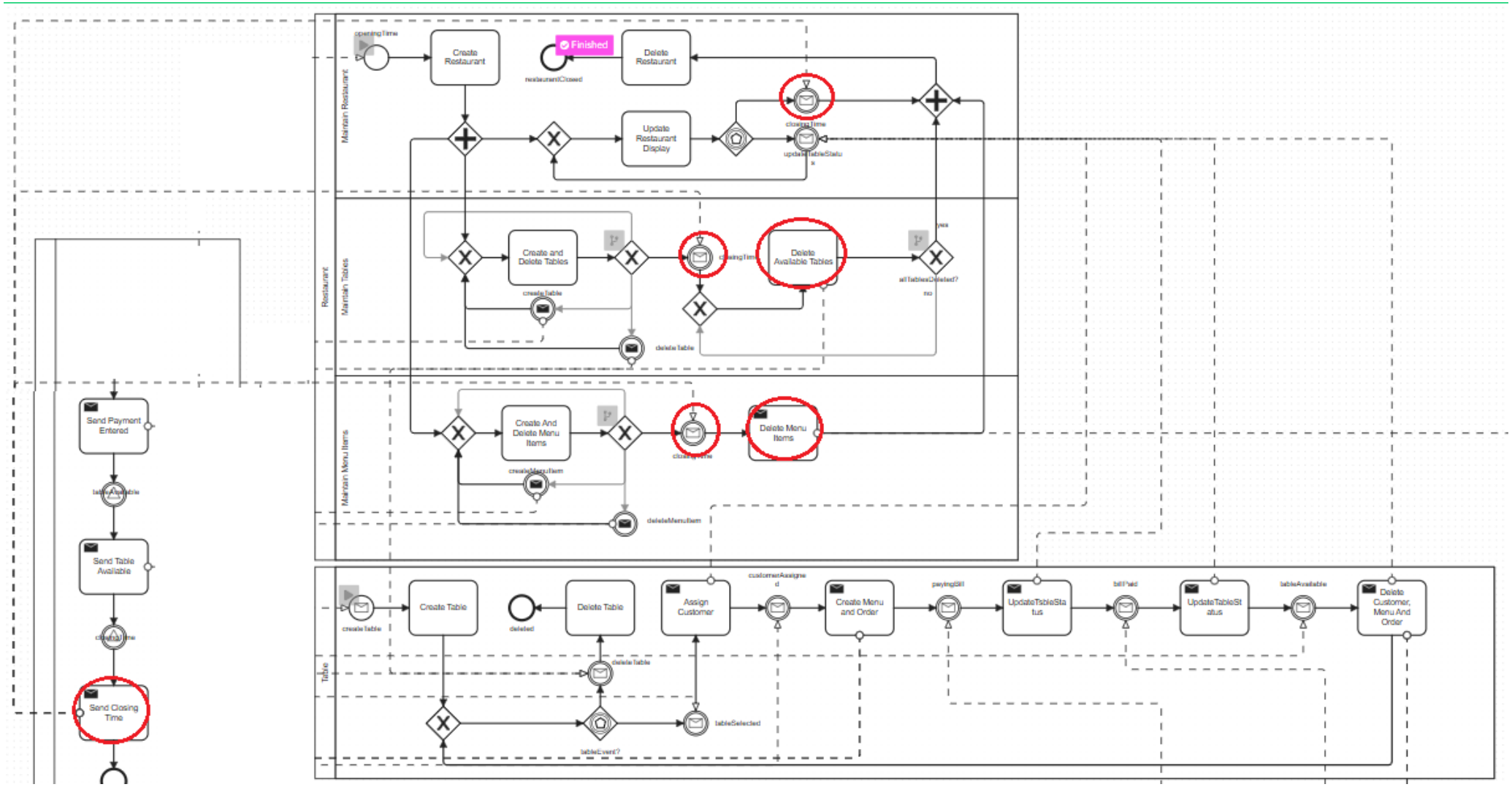
Customer Enters An Accepted Payment



Server Makes The Table Available



Restaurant Is Closing



Summary

- This work demonstrated how to model a business process with an system, and then simulate the execution of that system in order to verify the business vision for a future state of the process.
- The simulation contains no functional processing, (such as calculating the number of tables and menu items), nor does it connect to a user interface.
- Future steps will add functionality to the tasks and include user interfaces connected to tasks as forms.
- Add task times and object instances in order to analyze performance and resource needs.
- For future research: Investigate how to add data and build the solution using Camunda.

References

- Analysis Through Pictures – Describes the object-oriented analysis and design process used to create class diagrams.
- BPMN – The Object Management Group Business Process Model and Notation
- Visual Paradigm – UML and BPMN modeling tool.
- Camunda – BPMN development tool.
- Token Simulator – Simulation plugin for Camunda.