COMP 3004

Group x – Assignment 4 – Iteration 2

Introduction

The following document defines the design of a store simulation system, from the point of view of the simulation itself. The simulation models customers, employees and a manager acting and interacting with each other and with the fixed store and service points. The document considers three primary scenarios, one from the perspective of each actor: shopping, managing, and working a shift.

For this second and final iteration, our system supports the following functionality:

- managing and logging of events
- opening and closing of the store
- generating customers, both automatically at intervals and manually
- queueing of customers
- multiple queue selection strategies for customers
- processing of customers
- opening and closing of service points
- multiple management strategies for the store manager.
- reuse of employees when a service point closes and reopens

Design Overview

Introduction

When designing our simulation, we had several important questions to answer. As an event-driven simulation with multiple concurrent actors, one of the largest questions was how we handled time. Additionally, we had several other subsystems to worry about: logging, the user interface, and of course the actors and various other objects in the simulation itself. Our design attempts to separate all of these various subsystems, so that that any of the different pieces can be replaced with minimal effect on the rest of the system. The simulation core could, in principle, be used to run an entirely different type of simulation by changing only the actor and prop classes.

Time

The largest question we had to tackle in this design was our handling of time. Since the simulation contained multiple concurrent actors and events, some method would be needed to arbitrarily serialize things for logging purposes. We considered two main methods of implementing time.

The first method we considered was using multiple threads, one thread per actor, with the appropriate synchronization mechanisms (mutexes, etc). The operating system's thread scheduler would provide effective serialization, as only one thread would be able to hold the core mutex at a given time. While this design most accurately reflected the reality we were trying to simulate, and would also have been most efficient on multi-core CPUs, it was discarded for its complexity and potential for error.

The second method we considered, and the one we eventually chose, was to use a single priority queue for all events, prioritized based on time. The implementation of the priority queue would arbitrarily serialize simultaneous events by deciding which event with the same time of occurrence (and thus the same priority) was at the front of the queue. The simulation would simply take one event off the queue, process it, and repeat.

Actors and Props

The other large decision we had to make while designing our system was exactly what was considered an actor. The subjects of our three use cases obviously counted: customer, manager and employee, but what about the store itself? What about the service points? What about the queues? These elements were clearly deserving of objects of some sort, but those objects were not necessarily actors per se. After all, the store didn't actually act, it simply provided resources and state for other actors to use.

The design we eventually chose was to have a second type of objects, known as props. Props, unlike actors, are not capable of processing events in the simulation, however they can have other properties and methods available to be used by actors when needed. Additionally, while an event in the simulation must specify the actor to whom the event occurs, it may also specify a prop for the actor to retrieve and use during the processing of the event. In this way it is possible for an event 'enterServicePoint' to have the customer as the associated actor, and the service point itself as the associated prop.

Patterns

While designing our system, we decided to use several design patterns. Firstly, we use the Facade Pattern to abstract away the logging subsystem. This allows us to switch between multiple different types of logging with no changes to the rest of the program.

We also use the Singleton Pattern in several locations. Our Manager and Store objects, and the Simulator itself are all singletons, as having more than one of any of them at once makes no sense in our program. The Strategy Pattern also shows up a few times. The customer's service point selection algorithm, the queueing algorithm, and the Manager's decision algorithm are all abstracted away as strategies, so that slotting new algorithms into these locations is easy.

The final pattern we use is the Observer Pattern. Customers can be observers of the store if they are waiting for it to open before they start shopping.

Requirement Categories

Abbrev.	Name
LOG	Event Logging
SIM	Simulator Core Behaviour
PARAM	User-Specifiable Simulation Paramters

Functional Requirements

ID	Description	Motivation	Iteration
FR-LOG-01	The system must log all relevant events of the simulation.	Iter 1	1
FR-LOG-02	The log of each event must contain the time and nature of the event, as well as any relevant context.	Iter 1	1
FR-SIM-03	The simulation must permit the opening and closing of the store.	Iter 1	1
FR-SIM-04	The simulation must support automatic random generation of customers entering the store.	Iter 1	1
FR-PARAM-05	The simulation must allow the user to manually generate customers entering the store with a specified number of items at a specified point in time.	Iter 1 A Final Word about the Requirements	1
FR-SIM-06	Customers must spend an amount of time 'shopping' proportional to the number of items they wish to purchase.	G-C-01	1
FR-SIM-07	After a customer is finished 'shopping', they must be processed at a service point and leave the store.	G-C-01	1
FR-PARAM-08	The simulation must allow the user to control the minimum number of open service points.	C-04	2
FR-PARAM-09	The simulation must allow the user to control the maximum number of open service points.	C-01	2
FR-SIM-10	Before being processed, a customer must join the collection of customers looking for a service point.	G-C-01	1
FR-SIM-11	Once a customer looking for a service point has been chosen to proceed (by, for example, reaching the front of the queue), they must select a service point to be processed at.	G-C-02 A Final Word about the Requirements	1

ID	Description	Motivation	Iteration
FR-SIM-12	Once a customer has selected a service point, they must join the collection of customers waiting to be processed by that service point.	G-C-03	1
FR-SIM-13	Once a customer waiting at a service point has been chosen to proceed (by, for example, reaching the front of the queue), they must enter the service point to be processed.	G-C-06	1
FR-SIM-14	Processing a customer at a service point must take time proportional to the number of items the customer has chosen to purchase.	G-C-05	1
FR-SIM-15	Once a customer's purchases have been processed, they must take a constant amount of time to make payment arrangements.	G-C-05	1
FR-SIM-16	Once a customer has made payment arrangements, they must leave the service point and the store.	G-C-05	1
FR-PARAM-17	The user must be able to choose which method customers use to select a service point.	Iter 2	2
FR-SIM-18	The simulation manager must be able to open and close service points based on the current state of the store.	Iter 2	2
FR-PARAM-19	The user must be able to choose which method the manager uses to decide when to open/close service points.	Iter 2	2
FR-PARAM-20	The user must be able to choose which method is used to determine which customer in a collection proceeds next.	While the obvious choice (and the one suggested by the assignment) is a queue, it's not always that simple. Having worked as a cashier at a grocery store, free-for-alls can happen too.	2+
FR-SIM-21	The simulation must support service points with different usage restrictions and timing behaviours.	A Final Word about the Requirements	2+
FR-SIM-22	The simulation must support service points with multiple employees.	A Final Word about the Requirements	2+

ID	Description	Motivation	Iteration
FR-PARAM-23	The user must be able to choose which management strategy is used. Note: Not the same as FR-PARAM-19. That specifies choosing the manager's strategy, this specifies if there even is a manager.	Management decisions could be distributed across cashiers rather than having a central manager, for example.	2+
FR-SIM-24	Before the store can be closed, all the customers in the store must be processed and leave the store.	G-M-10	1
FR-LOG-25	Employees must record and calculate time spent working and idle during their shift	S-C-01	1
FR-SIM-26	Once a service point has been chosen to close, it must process all customers already in it's queue before the employee(s) are free to go.	G-M-5	2
FR-SIM-27	The store and a fixed number of service points must be opened before customers can be allowed in and processed.	G-M-3	1
FR-SIM-28	The queue of a particular service point may be limited in size.	WebCT Announcement. For iteration 1, they are to hold only 2 customers (not counting the one currently being processed).	1

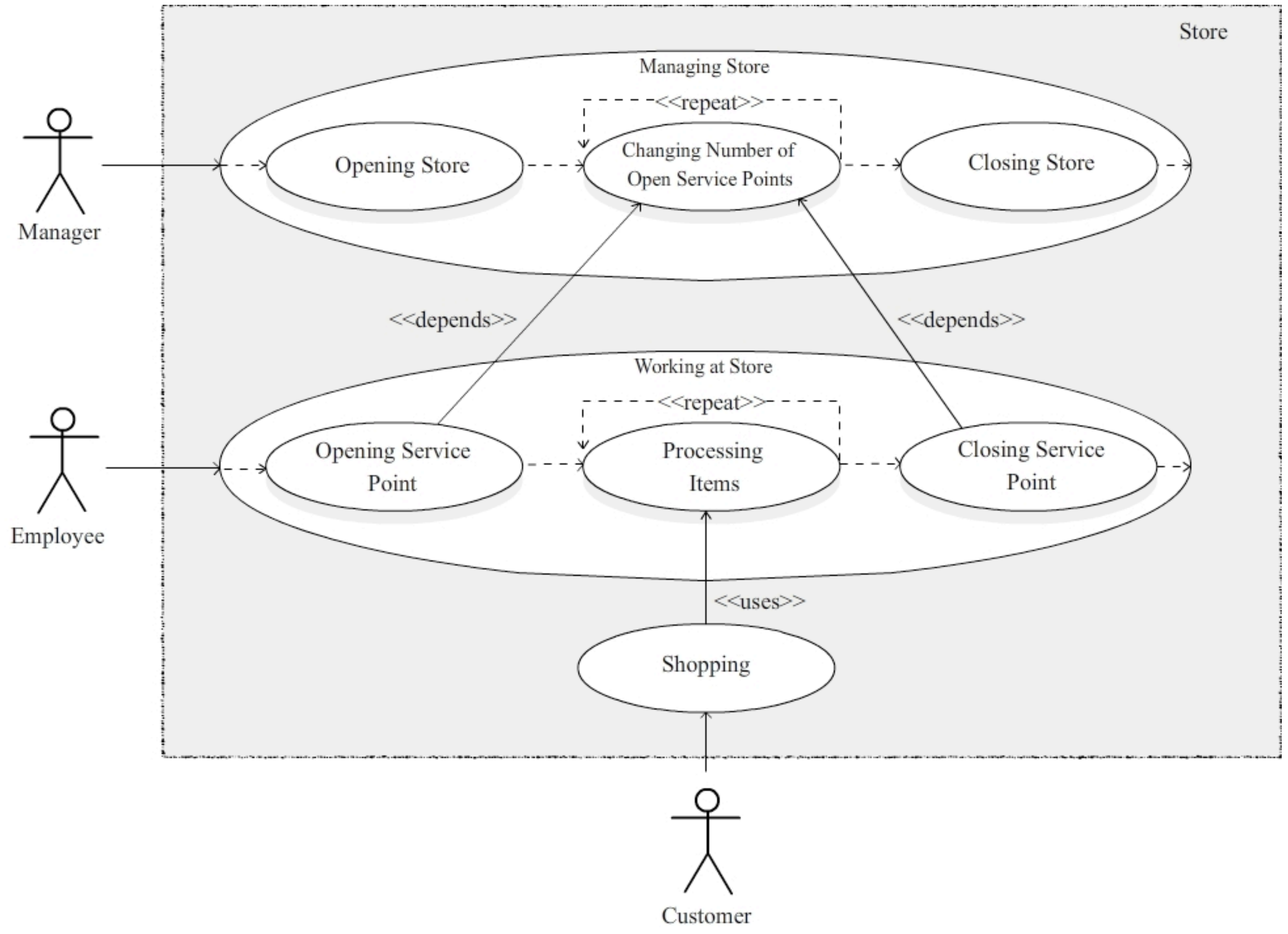
Non-Functional Requirements

ID	Description	Motivation
NFR-SIM-01	The customer wishes to spend a minimal amount of time waiting.	S-C-01
NFR-SIM-02	The managers wishes to minimize the time cashiers spend idle.	S-M-01

Assumptions

ID	Description	Reason
AS-SIM-01	Before leaving the store, a customer must purchase at least one item.	A customer who purchases no items is never waiting and uses no employee time, so permitting this case would complicate the model for no benefit.

Use Case Diagram



Use Cases

UC-01	Shopping	Traceability
Summary	Customer purchases items and is processed at a service point.	
External Actors	Customer, Employee	
Triggering Event	Customer enters store.	
Pre-Conditions	Store is open and a fixed number of service points are open.	FR-SIM-27
Main Sequence	**1.** Customer enters store and spend an amount of time 'shopping' proportional to the number of items they wish to purchase.	FR-SIM-06
	2. Customer joins the collection of customers looking for a service point (in future: 'seeking queue') with at least one item.	FR-SIM-07 FR-SIM-10 AS-SIM-01
	3. When the customer reaches the front of the seeking queue, they wait until at least one service point queue is not full, and then they select a service point to be processed at.	FR-SIM-28 FR-SIM-11 NFR-SIM-01
	4. Customer joins the collection of customers waiting to be processed by the service point that they chose in Step 3 (in future: 'processing queue').	FR-SIM-12
	5. When the customer reaches the front of the processing queue, they enter the service point to be processed.	FR-SIM-13
	6. Customer is processed by the employee(s) at the service point for an amount of time proportional to the number of items the customer has chosen to purchase and additionally a fixed constant time to make payment arrangements.	FR-SIM-14 FR-SIM-15
	7. Customer leaves the service point and exits the store	FR-SIM-16
Result	Customer exits the store	
Post-Condition	Customer is no longer in the store	

UC-02	Managing Store	Traceability
Summary	Manager manages the store.	
External Actors	Manager, Employees, Customers	
Triggering Event	Manager decides that it is time for the store to open.	
Pre-Conditions	Store is closed.	
Main Sequence	**1.** Manager requests that a fixed number of service points are to be opened.	FR-SIM-18
	2. Manager verifies that the requested number of service points have been opened.	FR-SIM-27
	3. Manager opens store.	FR-SIM-03
	4. Manager checks the arrival rate of customers waiting for a service point.	FR-SIM-10

	5. Manager requests that service points are opened or closed according to the management strategy.	FR-SIM-18 FR-PARAM-19 NFR-SIM-02
	6. Repeat to step 4 after a period of time.	
	7. Manager marks the store as closed so that new customers do not enter.	
	8. Manager ensures that all customers have been processed and have left the store.	FR-SIM-24
	9. Manager closes all open service points.	FR-SIM-26
	10. Manager closes store.	FR-SIM-03
Result	No customers and employees are in the store, the store and all service points are closed.	
Post-Condition	Store is closed.	
Alternatives	**UC-02-ALT-01:** If the manager wishes to close service points, but the minimum number is open, then nothing happens.	FR-PARAM-08
	UC-02-ALT-02: If the manager wishes to open service points, but the maximum number is open, then nothing happens.	FR-PARAM-09
	UC-02-ALT-03: When the store needs to close, skip step 6.	

UC-03	**Working at Store**	**Traceability**
Summary	An employee works at the store.	
External Actors	Employee, Customers	
Triggering Event	The manager requests that the employee open a service point.	FR-SIM-18
Pre-Conditions	- The service point is closed. - The employee is not already working somewhere else.	
Main Sequence	**1.** The employee records the beginning of their shift in order to be able to calculate total time working and idle.	FR-LOG-25
	2. The employee marks the service point as open.	FR-SIM-18
	3. The employee processes customers until the manager requests the service point be closed.	FR-SIM-14 FR-SIM-15
	4. The employee marks the service point as closed.	FR-SIM-18
	5. The employee processes all remaining customers at the service point.	FR-SIM-26
	6. The employee records the end of their shift and calculates their final time working and idle.	FR-LOG-25
Result	Customers have been processed.	
Post-Condition	The service point is closed.	
Alternatives	**UC-03-ALT-01:** If the service point is already idle when asked to close, step 5 will be skipped.	

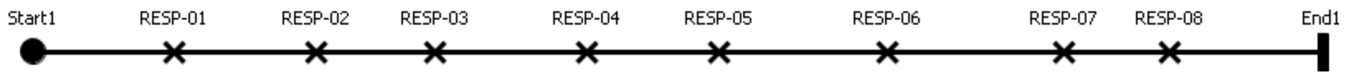
Responsibilities

ID	**Responsibility**	**Use Case Steps**

ID	Responsibility	Use Case Steps
RESP-01	Customer shops.	UC-01-1
RESP-02	Customer enters queue to seek service point.	UC-01-2
RESP-03	Customer chooses available service point.	UC-01-3
RESP-04	Customer joins queue of selected service point.	UC-01-4
RESP-05	Customer enters service point.	UC-01-5
RESP-06	Customer's items are processed.	UC-01-6, UC-03-3, UC-03-5
RESP-07	Customer makes payment.	UC-01-6, UC-03-3, UC-03-5
RESP-08	Customer leaves service point.	UC-01-7
RESP-09	Service point(s) are opened.	UC-02-1, UC-02-5, UC-03-2
RESP-10	Manager verifies that service points are opened.	UC-02-2
RESP-11	Store is marked as open.	UC-02-3
RESP-12	Manager checks the arrival rate of customers waiting for a service point.	UC-02-4
RESP-13	Service point(s) are closed.	UC-02-5, UC-02-9, UC-03-4
RESP-14	Store is marked as closed.	UC-02-7
RESP-15	Manager ensures all customers have been processed and left the store.	UC-02-8
RESP-16	Store is closed.	UC-02-10
RESP-17	Beginning of shift is recorded.	UC-03-1
RESP-18	End of shift recorded.	UC-03-6
RESP-19	Shift statistics calculated.	UC-03-6

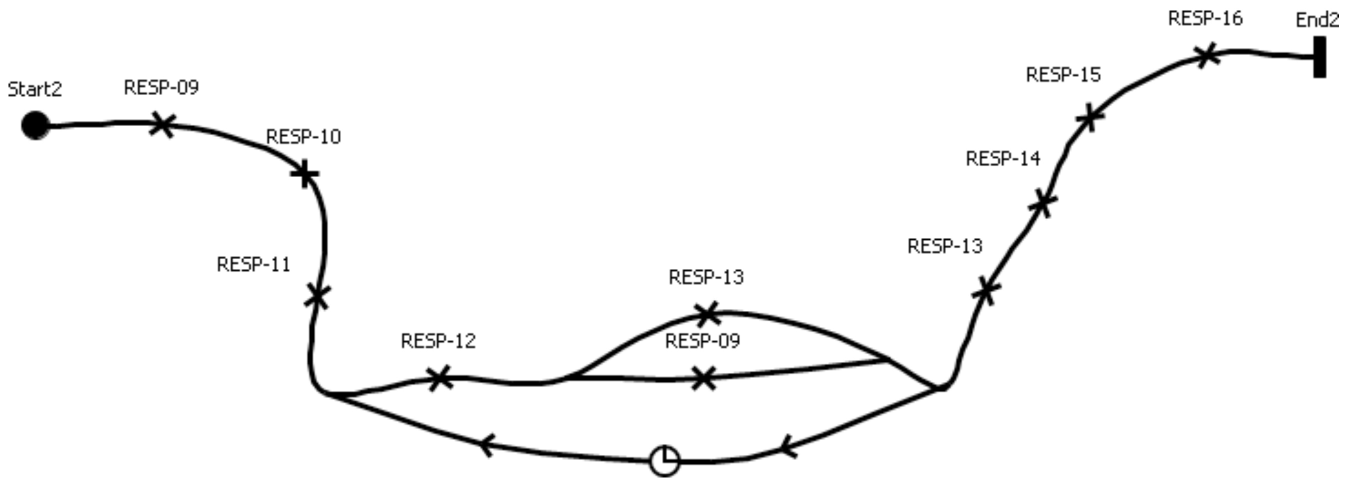
Unbound Use Case Maps

UC-01



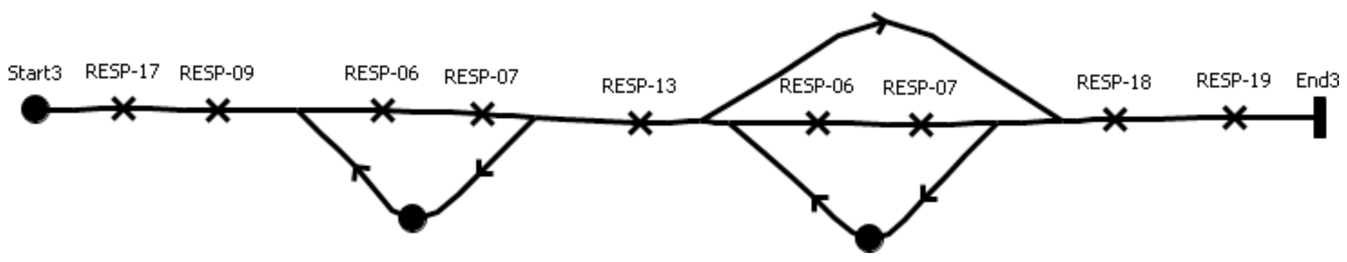
Start1	Customer enters store.
End1	Customer exits store.

UC-02



Start2	Manager decides to open the store.
End2	Manager closes store.

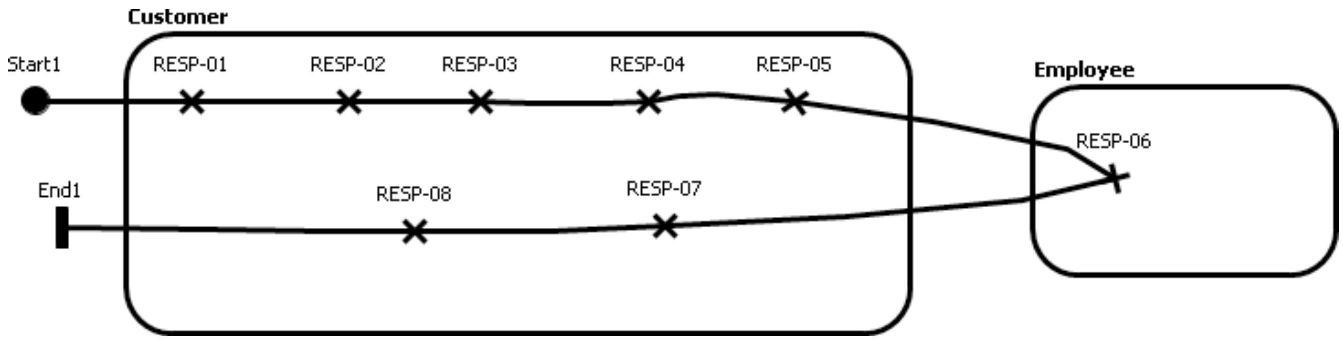
UC-03



Start3	Manager asks employee to work.
End3	Employee's shift is over.

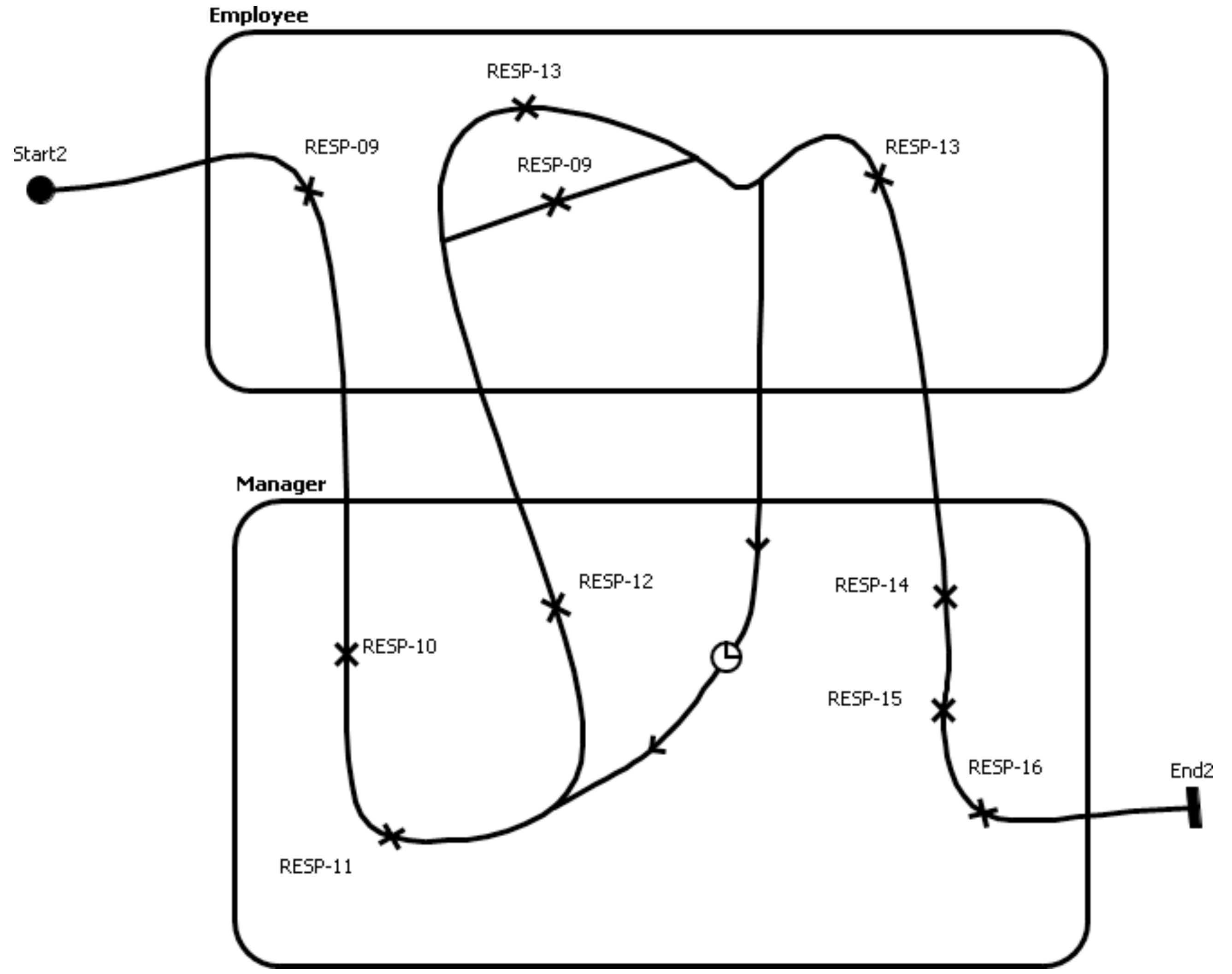
Bound Use Case Maps

UC-01



Start1	Customer enters store.
End1	Customer exits store.

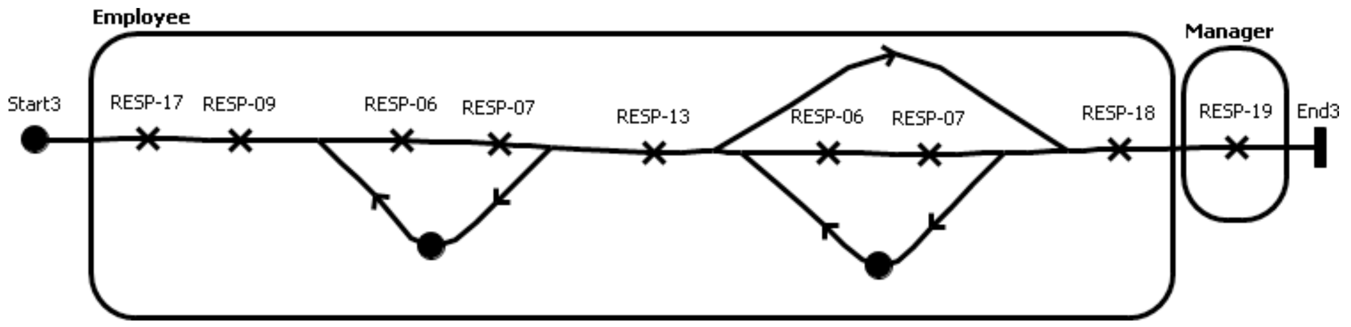
UC-02



Start2 Manager decides to open the store.

End2 Manager closes store.

UC-03

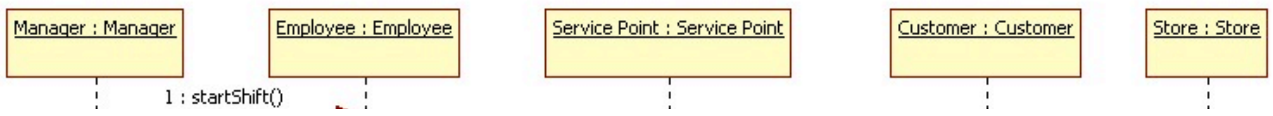


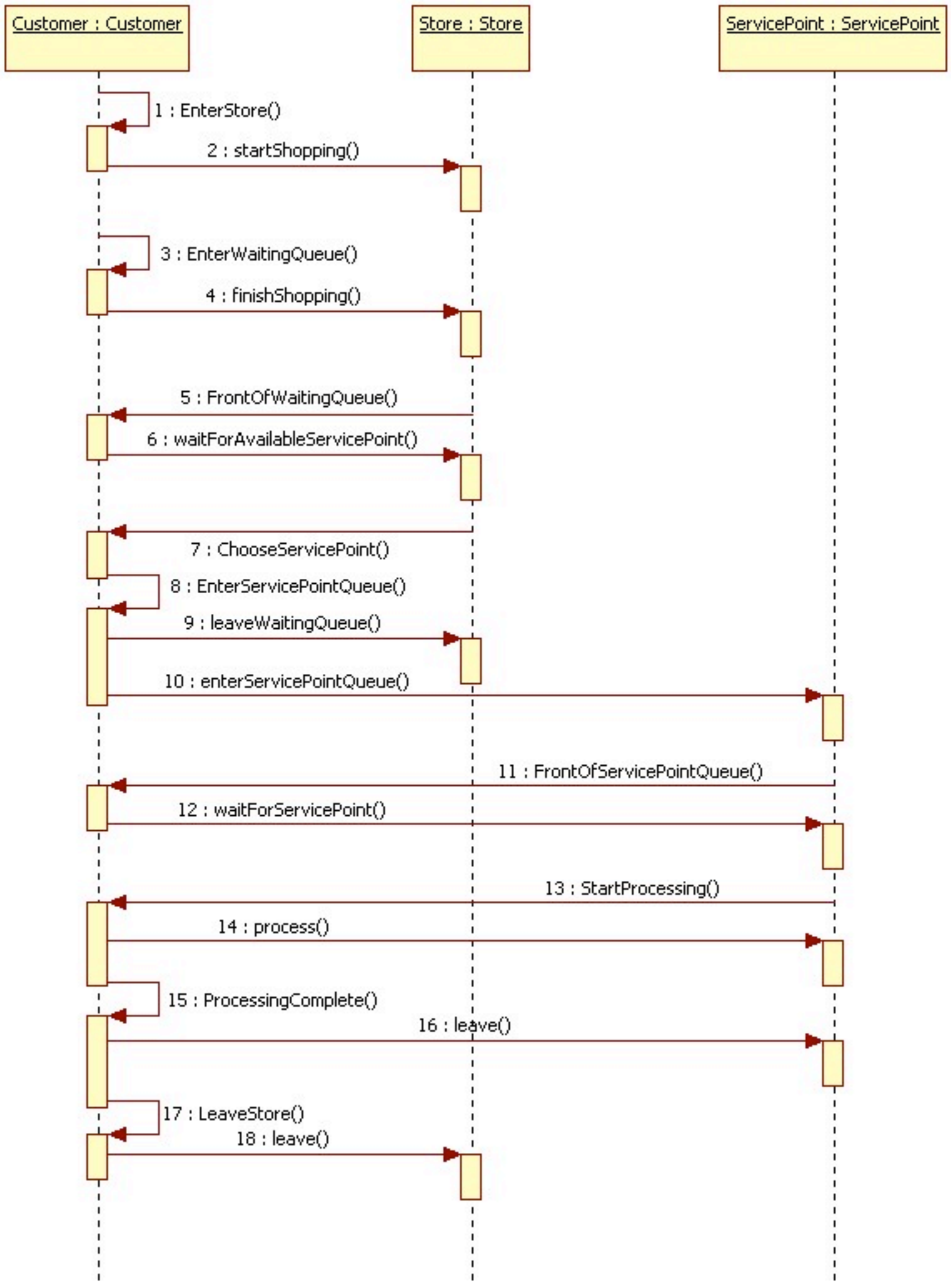
Start3 Manager asks employee to work.

End3 Employee's shift is over.

Interaction Diagrams

High-Level Interaction Diagram

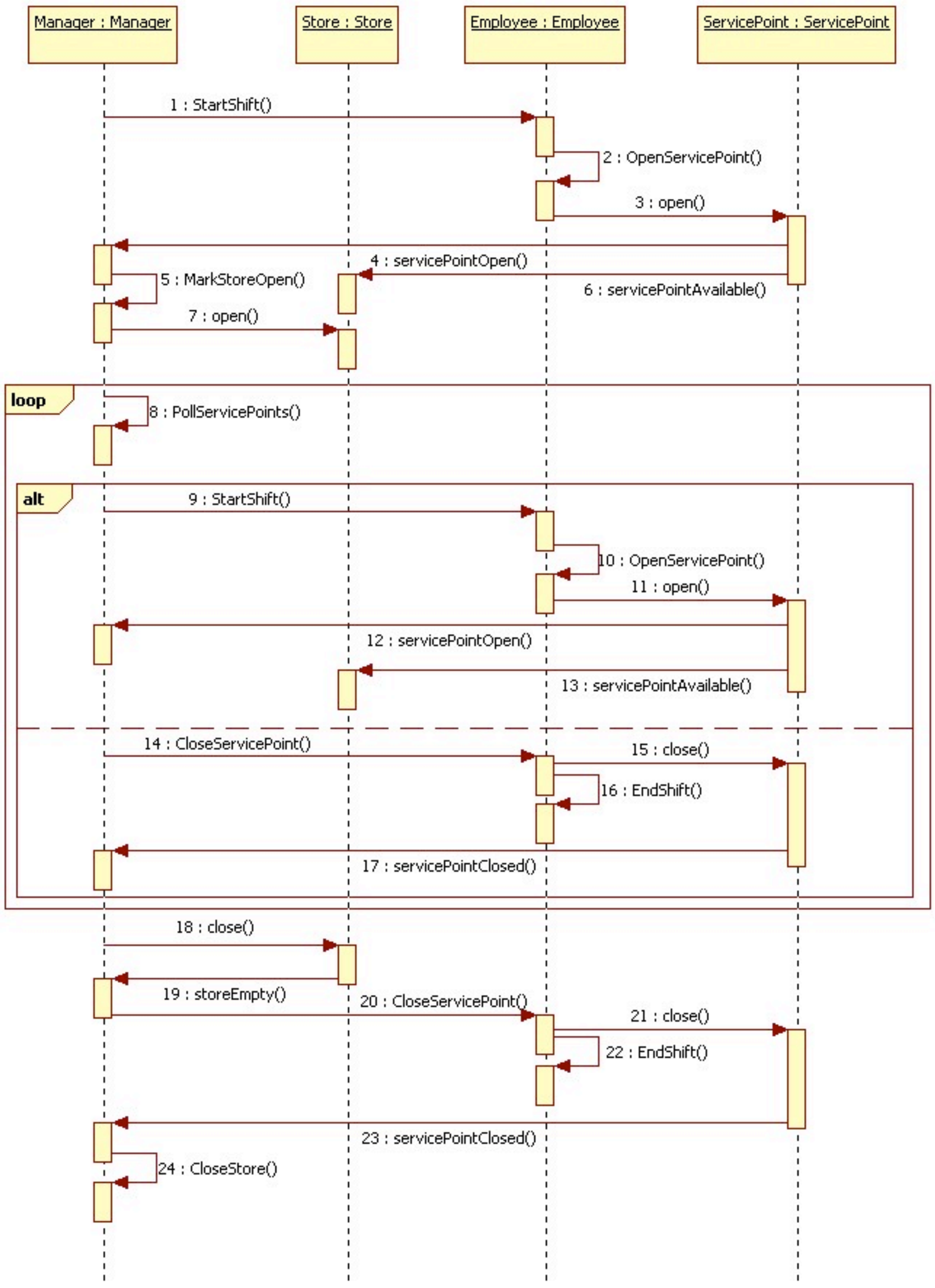




UC

UC-
0
2

UC-
0
3


Manager : Manager
Store : Store
Employee : Employee
ServicePoint : ServicePoint
1 : StartShift()
2 : OpenServicePoint()
3 : open()
4 : servicePointOpen()
5 : MarkStoreOpen()
6 : servicePointAvailable()
7 : open()
loop
8 : PollServicePoints()
alt
9 : StartShift()
10 : OpenServicePoint()
11 : open()
12 : servicePointOpen()
13 : servicePointAvailable()
14 : CloseServicePoint()
15 : close()
16 : EndShift()
17 : servicePointClosed()
18 : close()
19 : storeEmpty()
20 : CloseServicePoint()
21 : close()
22 : EndShift()
23 : servicePointClosed()
24 : CloseStore()

Object Specifications

Customer

Overview

Name	ID	Description	Inheritance
Customer	ACT-01	Shops at the store, purchasing one or more items and making payment before leaving.	None

Responsibilities

ID	Responsibilities	Collaborators	Traceability
RESP-01	Customer shops.		UC-01-1
RESP-02	Customer enters queue to seek service point.		UC-01-2
RESP-03	Customer chooses available service point.	Employee	UC-01-3
RESP-04	Customer joins queue of selected service point.		UC-01-4
RESP-05	Customer enters service point.		UC-01-5
RESP-07	Customer makes payment.	Employee	UC-01-6, UC-03-3, UC-03-5
RESP-08	Customer leaves service point.		UC-01-7

Data Members

Variable Name	Type	Status	Procedures
numberOfItems	uint	Read-Only	
entranceTime	Time	Read-Only	
waitTimeStart	Time	Read/Write	
waitTimeTotal	Time	Read/Write	
sp	ServicePoint	Read/Write	

Procedures

Procedure Name	Return Type	Parameter Name	Parameter Type	Responsibilities
processEvent	void	e	Event	Makes an appropriate action based on event e.
myString	string	e	Event	Returns additional useful string to log based on event e.

Manager

Overview

Name	ID	Description	Inheritance

Name	ID	Description	Inheritance
Manager	ACT-02	Responsible for opening and closing the store as well as using effective management strategies to decide which service points to request for open or close to better serve customers.	None

Responsibilities

ID	Responsibilities	Collaborators	Traceability
RESP-10	Manager verifies that service points are opened.		UC-02-2
RESP-11	Store is marked as open.		UC-02-3
RESP-12	Manager checks the arrival rate of customers waiting for a service point.		UC-02-4
RESP-14	Store is marked as closed.		UC-02-7
RESP-15	Manager ensures all customers have been processed and left the store.	Employees, Customers	UC-02-8
RESP-16	Store is closed.		UC-02-10
RESP-19	Shift statistics calculated.		UC-03-6

Data Members

Variable Name	Type	Status	Procedures
myInstance	Manager		
initialOpenServicePoints	static int		
minimumServicePoints	static int		
maximumServicePoints	static int		
pollingInterval	static uint		
totalIdleTime	uint		
totalShifts	uint		
empPool	EmployeePool		
activeEmployees	HashSet<Employee>		

Procedures

Procedure Name	Return Type	Parameter Name	Parameter Type	Responsibilities
getInstance	Manager			Returns the instance of Manager for the singleton pattern.
servicePointOpen	void	e	Employee	Increases the number of service points known to be open. If sufficient service points are open and the store is marked as closed, marks the store as open.

Procedure Name	Return Type	Parameter Name	Parameter Type	Responsibilities
servicePointClosed	void	e	Employee	Decreases the number of service points known to be open. If there are no service points left, closes the store.
storeEmpty	void			Closes all service points still open.
openServicePoints	void	change	int	Opens 'change' number of new service points.
closeServicePoints	void	change	int	Closes 'change' number of open service points.
processEvent	void	e	Event	Makes an appropriate action based on event e.
myString	string	e	Event	Returns additional useful string to log based on event e.

Employee

Overview

Name	ID	Description	Inheritance
Employee	ACT-03	Works at store. Responsible for opening and closing service points, processing customers and directing customers to service points when requested by the store manager.	None

Responsibilities

ID	Responsibilities	Collaborators	Traceability
RESP-06	Customer's items are processed.	Customers	UC-01-6, UC-03-3, UC-03-5
RESP-07	Customer makes payment.	Customer	UC-01-6, UC-03-3, UC-03-5
RESP-09	Service point(s) are opened.		UC-02-1, UC-02-5, UC-03-2
RESP-13	Service point(s) are closed.		UC-02-5, UC-02-9, UC-03-4
RESP-17	Beginning of shift is recorded.		UC-03-1
RESP-18	End of shift recorded.		UC-03-6

Data Members

Variable Name	Type	Status	Procedures
sp	ServicePoint		
shiftStart	Time		
shiftLengths	List<uint>		
numItemsProcessed	uint		
customersProcessed	uint		
idleTime	uint		

Procedures

Procedure Name	Return Type	Parameter Name	Parameter Type	Responsibilities
updateNumItems	void	items	uint	Tracks total items processed by adding items to numItemsProcessed.
updateNumCustomers	void			Tracks total customers processed by adding one to customersProcessed.
processEvent	void	e	Event	Makes an appropriate action based on event e.
myString	string	e	Event	Returns additional useful string to log based on event e.

Testing

The logged output from all of our test cases can be found in the text files named 'test##.log' in the folder 'A4-Iter2-Team1-Tests' included with this document. A brief description of what each test demonstrates is included here.

Case 01

This case demonstrates all of the available configuration options except for customer creation.

Case 02

This case demonstrates the various customer creation options.

Case 03

This case demonstrates that customers can enter and leave the store simultaneously. The manager demonstrates the 'lazy' management strategy (where no service points are ever opened and closed beyond the defaults). Customer 1 demonstrates passing directly through the waiting queue when it is empty, and all customers demonstrate the 'simple' queue selection strategy (also known as the iteration 1 queue selection strategy).

Case 04

This case demonstrates the customers using the waiting queue to wait while all of the open service points are full. This can be seen when customer 25 reaches the front of the waiting queue at 8:55, but has to wait until 8:59:20 to choose a service point.

Case 05

This case demonstrates the use of the 'queue size' management strategy. Multiple simultaneous closures can be seen at 8:10, a single opening can be seen at 8:30, and a single closure can be seen at 8:40.

Case 06

This case demonstrates the 'arrival rate' management strategy and the 'simple' customer strategy.

Opening of a service point can be seen at 8:20 and every 5 min (manager polling interval) up until 8:35. Closing of a service point can be seen at 8:40, 8:45 and 8:50. This case also demonstrates that the store will remain open after closing time to process any remaining customers that are still waiting to be processed.

Case 07

This case demonstrates the 'fewest items' customer queue selection strategy. Service point 1 has 2 customers in its queue with a total of 209 items. Service point 2 has 2 customers in its queue with a total of 210 items. Customer 5 gets to the front of the waiting queue and chooses service point 1, as it has fewer total items.

Case 08

This case demonstrates the 'fewest customers' customer queue selection strategy. Note that the service point selected by each subsequent customer alternates between 1 and 2, demonstrating that the customers are selecting the service point with the fewest customers.