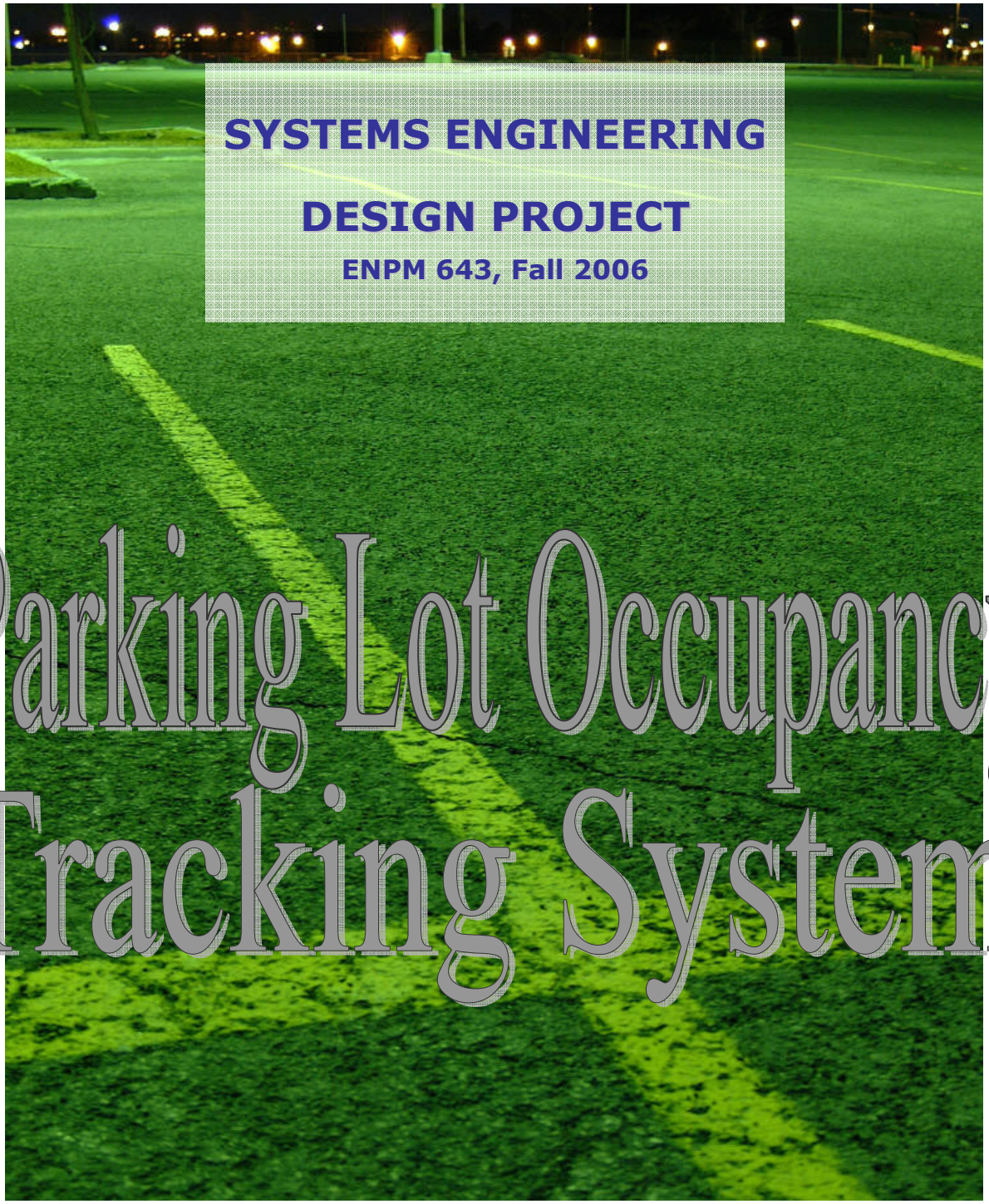


SYSTEMS ENGINEERING

DESIGN PROJECT

ENPM 643, Fall 2006

Parking Lot Occupancy Tracking System

Instructor	*Authors*	*ENPM643*
Dr. M Austin	Atul Mehta & Felipe Leite	Fall 2006

TABLE OF CONTENTS

1 OVERVIEW OF THE PROJECT

1.1 Purpose

The goal of this project was to take a parking lot control system from requirement generation, through model generation and validation. This document describes the process taken a presents the resulting data.

1.2 Roles and Responsibilities

The project team is formed by the following students: Atul Mehta and Felipe Leite. Since all team members have the same experience and expertise designing systems, the work was divided equally and assigned arbitrarily.

1.3 Introduction

For our project, we have decided to design a system that can be deployed in existing parking structures which would provide information about available parking spaces to drivers trying to access the facility. The system will include software, sensors and the networking components.

The installation of permanent sensors in each parking space will provide lot owners with constant and accurate information on parking lot occupancy. This allows them to keep the lot at full capacity and serve customers better. Figure 1 shows the building parking lot that we will be designing for. It has 72 parking spaces where 8 are handicap, 4 are for the building staff and the remaining 60 are available for regular lot users. In addition, a list of members provided to the system by the building owner will be used to grant or not access to the facility.

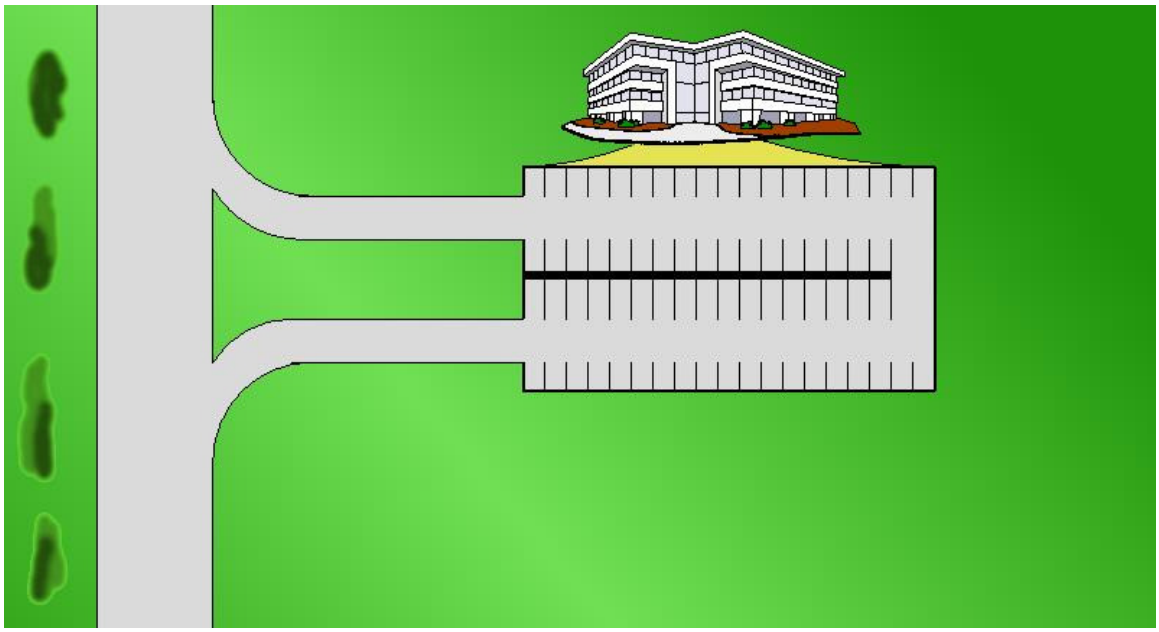


Figure 1 - Parking Lot Layout

The system will make use of electronic signs to give drivers, information regarding parking availability before they enter the facility. Once inside, color coded LED displays will lead motorists to the vacant spaces. Parking spaces can be made unavailable by the building manager bypassing the sensors. This and other configuration options will be made available through a GUI interface.

1.4 Definitions

1.4.1 Parking Lot

The parking lot consists of one entrance and 1 exit. There are 72 parking spaces with 8 being handicap and 4 being reserved for building staff.

1.4.2 Entrance

The entrance consists of a gate, a display showing the precise number of available parking spaces, a tag reader. The tag reader is activated as soon as the car is within range.

1.4.3 Exit

The exit consists of a gate and an induction loop that is behind the gate to detect when a car approaches the gate.

1.5 Assumptions and Limitations

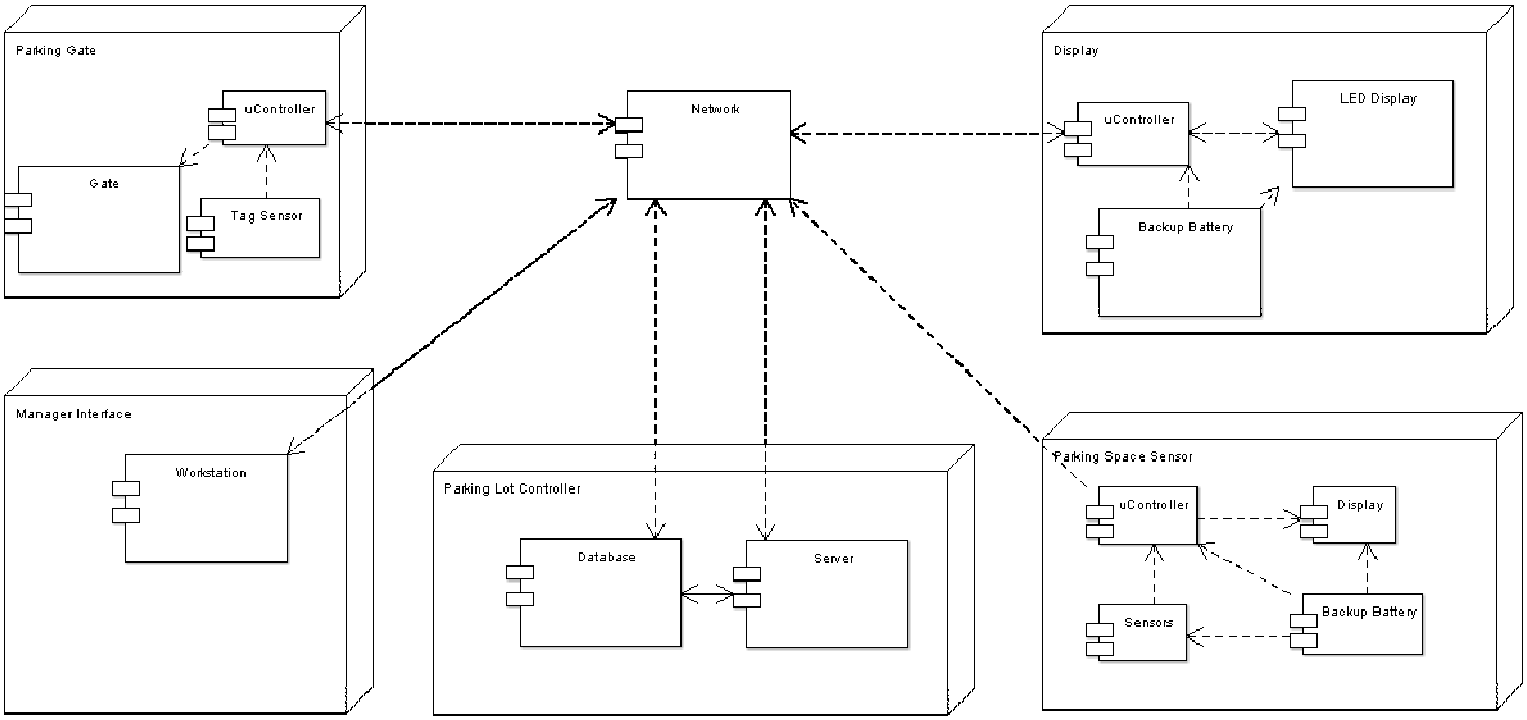
- Every parking space can be reached from any entrance.
- Every exit can be reached from each parking space.
- No entrances are convertible to exits and vice versa.
- Building manager assigns a unique tag to each authorized car.
- Tag number and vehicle information is stored in a database maintained by the building manager.
- Emergency situations (e.g. fire) will not be considered here.

1.6 Reference Material

[1] J Magee and J Kramer, "Concurrency State Models and JAVA Programming," John Wiley & Sons Ltd., 2nd Edition, 2006

2 UML MODELS

2.1 Deployment Diagram



The Parking Lot Occupancy Tracking system can be divided into sub-components as seen in the deployment diagram shown above. The Parking Lot Occupancy Tracking system consists of a parking gate interface, manager interface, parking lot controller system interface, parking sensor system interface, display interface and network interface.

Parking Gate Interface

This subsystem is triggered as soon as a commuter/car arrives near the parking gate. The scanner scans the tag. The unit controller in the interface checks the authorization of the commuter with the database of the system. If authorization confirmed the unit controller triggers the gate to open and let the commuter in the parking lot. If authorization fails, the gate is not opened.

Manager Interface

Manager interface system is an indirect component to the system. The owner/manager of the system plays the role of the manager interface. The system is automated and hence the only role of the manager during the running system is to start/stop, add/delete/edit details of commuters and generates report from the system database. The manager however has the privileges of changing the system configuration and display manually entered messages on the display board during maintenance or system troubleshooting.

Parking Lot Controller Interface

Parking lot controller interface consist of servers and a database that records all the events the system goes through the day. The controller receives and sends information to the unit controllers as well as displays the information on the main display board.

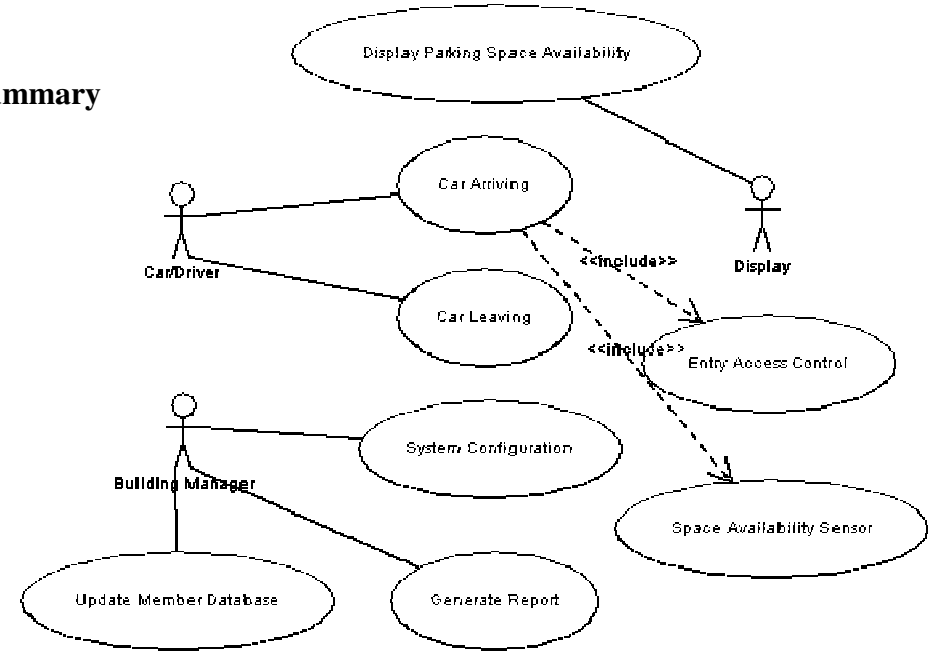
Parking Sensor System Interface

Parking sensor system interface consist of sensors, unit controllers, unit display boards and back-up battery for the system. This system is triggered when a car parks or leaves the parking lot. When a commuter parks the car in the parking lot, the sensor detects the action and sends information to the unit controller. The unit controller receives the information and triggers the unit display board to record the action and also send the information to the main controller interface to display the action on the main display board.

Display Interface

Display interface displays the status of the system. It has a unit controller, which records the information and displays the message through the display board. It also consists of a backup battery for the smooth running of the system.

2.2 Use Case Summary

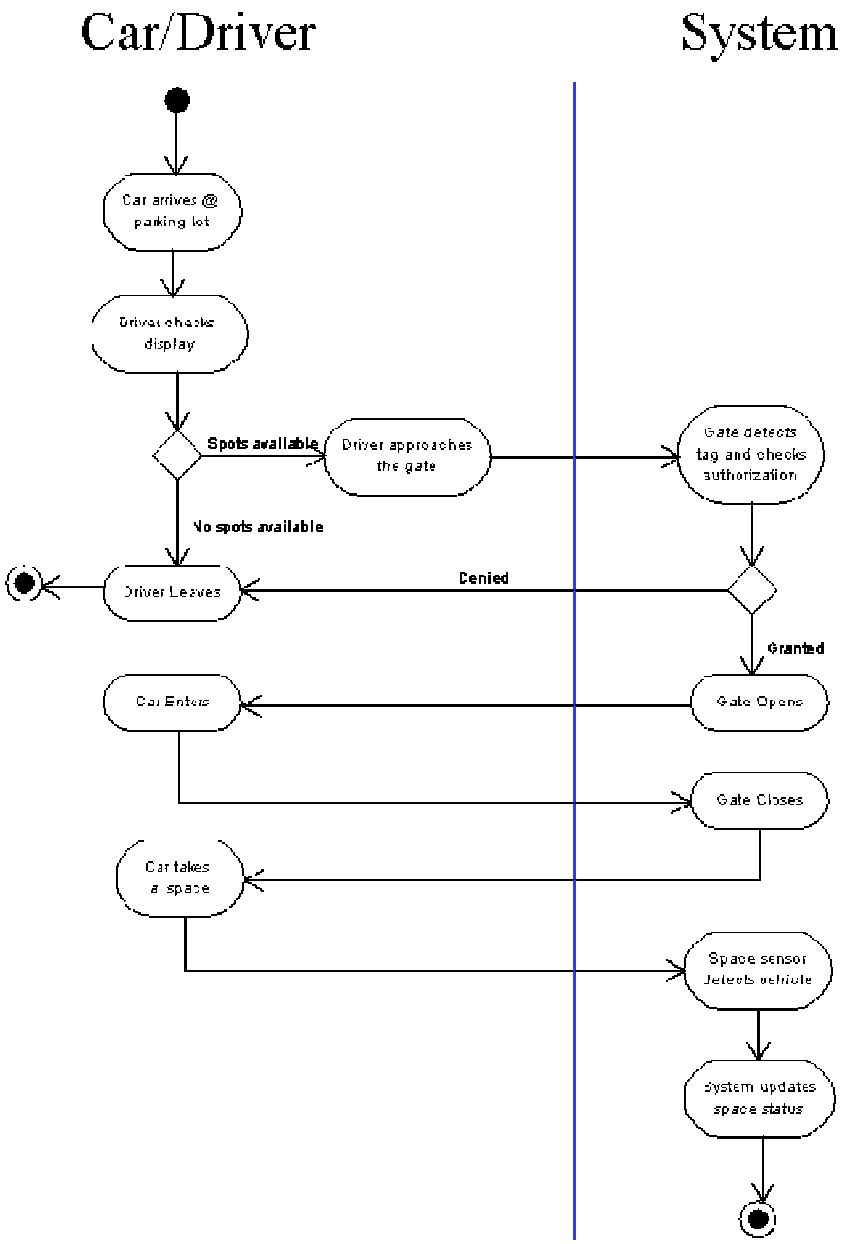


2.3 Car Arriving

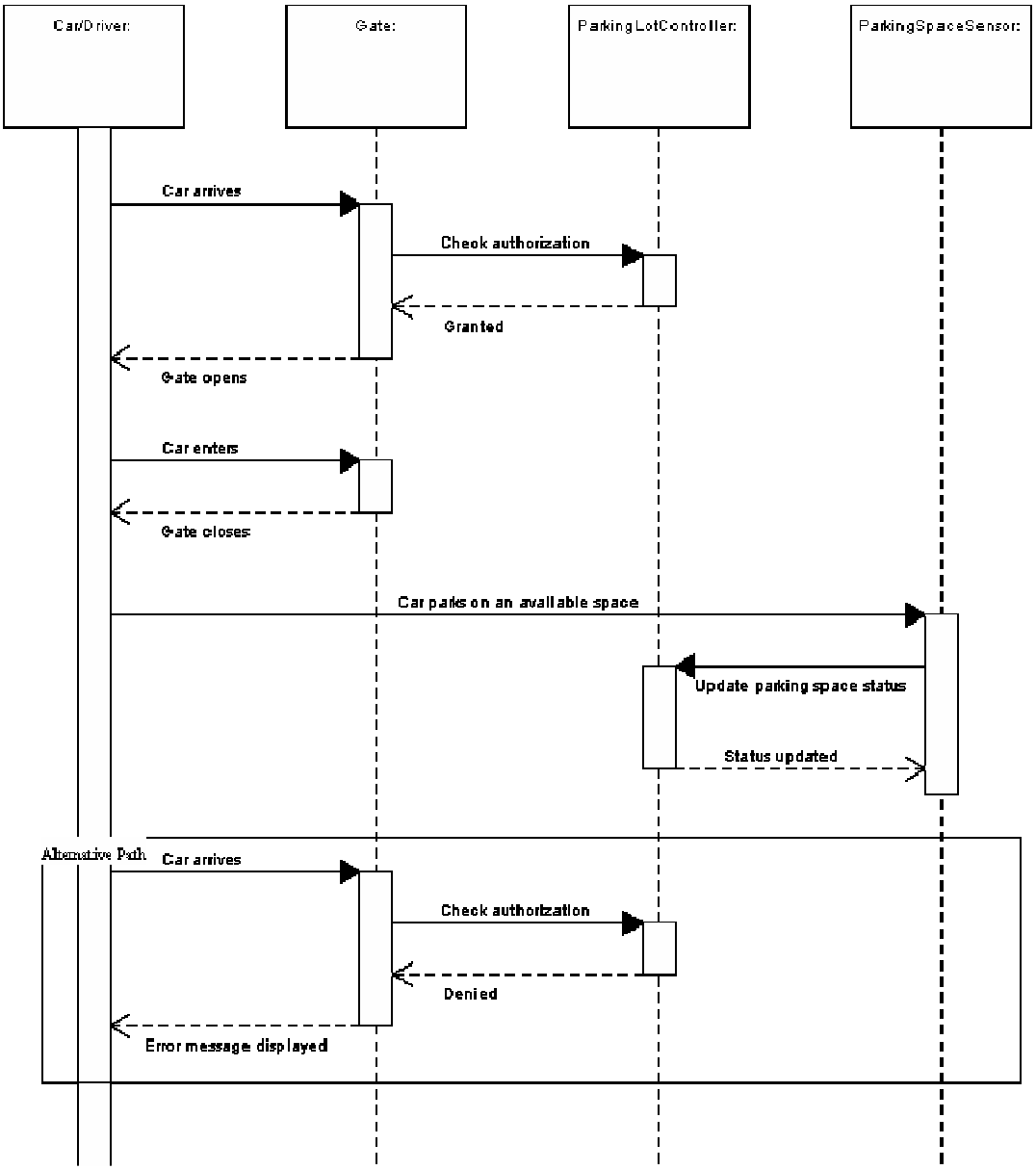
Use Case Narrative

Use-Case Name:	Car Arriving	
Primary Actor:	Car/Driver	
Other Actors:	None	
Description:	This use case describes the event of the system tracking when a car arrives at the parking lot	
Assumptions:	Parking Lot Open	
Precondition:	Car must have a parking lot tag	
Initiation/Trigger:	The use case is initiated when the car approaches the entrance gate	
Typical Course Of Events /Dialog:	**Actor Action**	**System Response**
	Step 1: Car approaches gate	**Step 2**: The system detects the car tag and checks with may database for authorization. **Step 3**: The car is authorized and the system responds by opening the gate.
	Step 4: The car drives through the gate.	**Step 5**: The system closes the gate once the car is through.
	Step 6: The car parks in an available space.	**Step 7**: The system detects car. **Step 8**: The space status is updated within the system.
Alternate Courses:	**Alt-Step 3**: If the car access is not authorized, an error message is displayed to inform the outcome that the access has been denied	
Conclusion:	This use case concludes when the parking space status is updated	
Post Condition:	Parking space status is updated from vacant to occupied	

Activity Diagram



Sequence Diagram

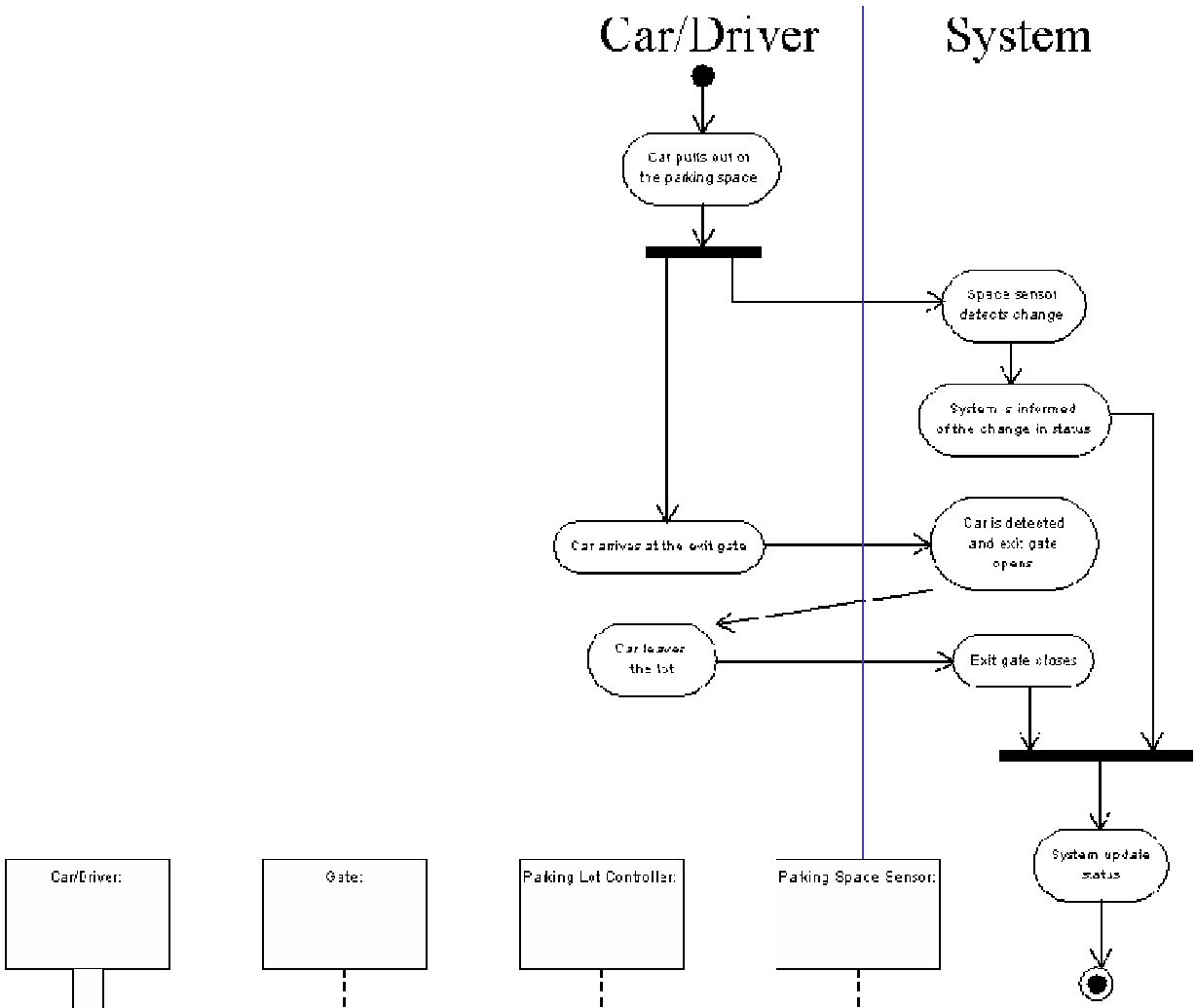


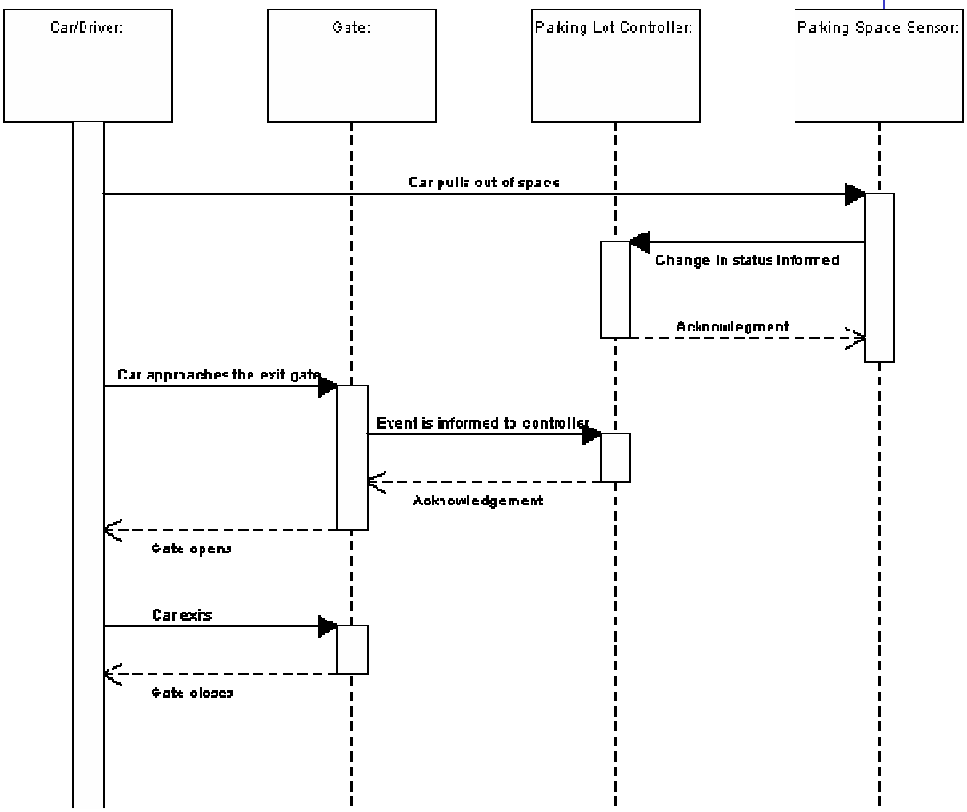
2.4 Car Leaving

Use Case Narrative

Use-Case Name:	Car Leaving	
Primary Actor:	Car/Driver	
Other Actors:	None	
Description:	This use case describes the event of the system tracking when a car leaves the parking lot	
Assumptions:	Parking Lot Open	
Precondition:	Car is inside the parking lot	
Initiation/Trigger:	The use case is initiated when the car approaches the exit gate	
Typical Course Of Events /Dialog:	**Actor Action**	**System Response**
	Step 1: Car approaches the exit gate	**Step 2**: The system detects the car by the exit gate
		Step 3: The system sends a signal for the exit gate to open
		Step 4: The exit gate opens
	Step 5: The car leaves the parking lot premises	**Step 6**: The exit gate closes
Alternate Courses:	N/A	
Conclusion:	This use case concludes when the exit gate closes	
Post Condition:	The car is no longer within the parking lot limits	

Activity& Sequence Diagrams



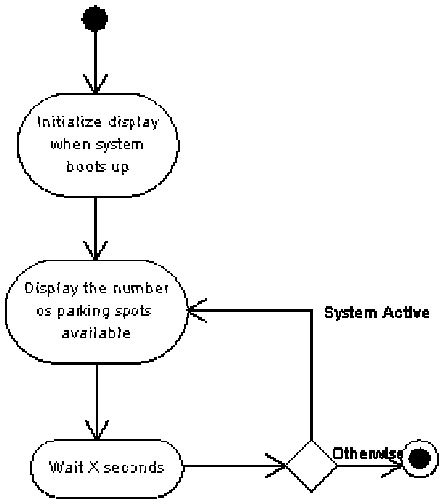


2.5 Display Parking Space Availability

Use Case Narrative

Use-Case Name:	Display Parking Space Availability	
Primary Actor:	Display	
Other Actors:	None	
Description:	This use case describes the event of the system updating the information displayed by the parking lot entrance	
Assumptions:	Parking lot open and display on	
Precondition:	None	
Initiation/Trigger:	The use case is initiated when the parking lot opens	
Typical Course Of Events /Dialog:	**Actor Action**	**System Response**
	Step 2: Display acknowledges initialization **Step 4**: Information is displayed	**Step 1**: The system initializes the display **Step 3**: The system sends information to the display **Step 5**: The system waits a preset amount of time and goes back to **Step 3**
Alternate Courses:	**Alt Step 5** If system disable the loop is terminated	
Conclusion:	This use case concludes when the system is disabled	
Post Condition:	The up-to-date information on available parking spaces is displayed	

Activity Diagram



2.6 System Privileges/Utilities

Use Case Narrative

<table>
<tr><td>Use-Case Name:</td><td colspan="3">System Privileges/Utilities</td></tr>
<tr><td>Primary Actor:</td><td colspan="3">Manager/Owner</td></tr>
<tr><td>Other Actors:</td><td colspan="3">None</td></tr>
<tr><td>Description:</td><td colspan="3">This use case describes the possible system priveleges available to the parking lot owner.</td></tr>
<tr><td>Assumptions:</td><td colspan="3">NA</td></tr>
<tr><td>Precondition:</td><td colspan="3">Manager must have the access password to the system</td></tr>
<tr><td>Initiation/Trigger:</td><td colspan="3">The manager enters the password to get access to the system.</td></tr>
<tr><td rowspan="5">Typical Course Of Events /Dialog:</td><td colspan="3">Actor Action</td></tr>
<tr><td colspan="3">Step 1: Owner enters password to get access to the system.</td></tr>
<tr><td colspan="3">Step 2: Owner is authorized and has a choice of three options. (a/b/c)

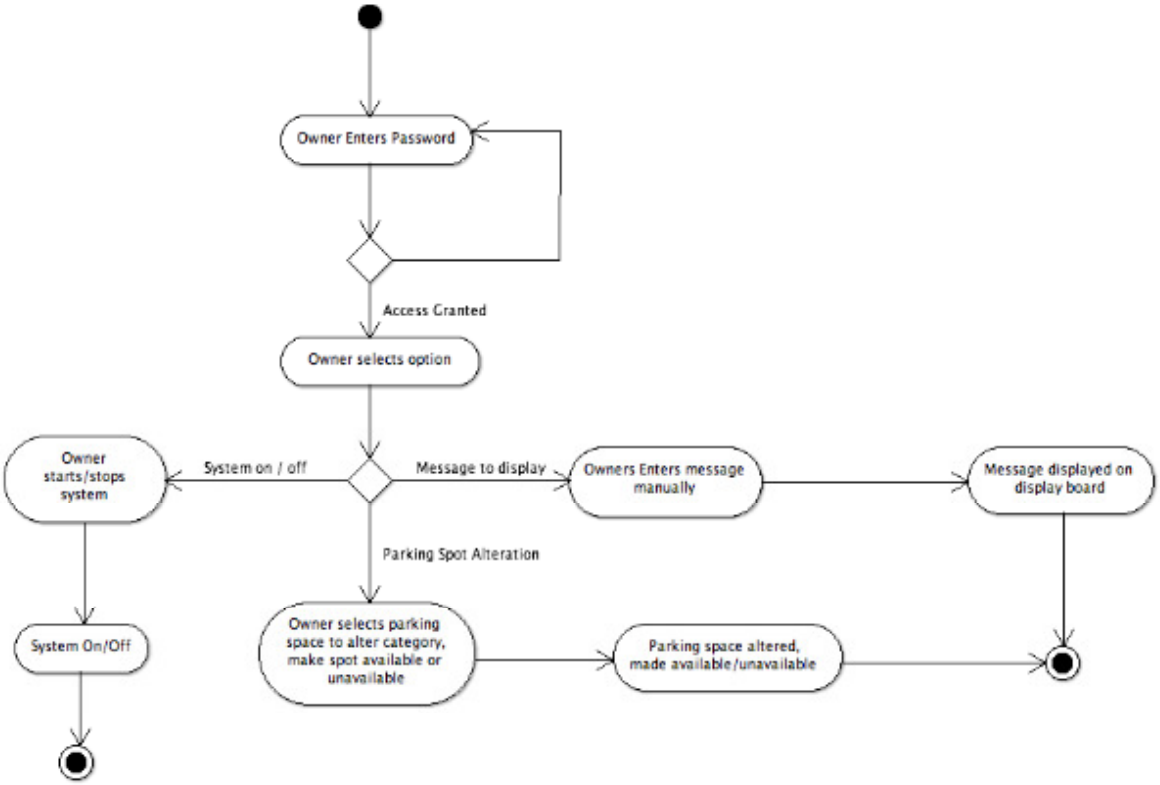
Step 3: The Owner selects the option. (a/b/c)</td></tr>
<tr><td>Step 4.a: Owner selects system on/off.

Step 5.a: System turned on/off</td><td>Step 4.b: Owner selects parking space to be altered, made available/unavailable

Step 5.b: Parking space is altered.</td><td>Step 4.c: Owner enters message manually for the display board.

Step 5.c: Only the entered message is displayed on the board.</td></tr>
<tr><td>Alternate Courses:</td><td colspan="3">Alt-Step 2: Owner is not authorized and is unable to take any action on the system.</td></tr>
<tr><td>Conclusion:</td><td colspan="3">This use case concluded when system turned on/off or space altered or message is displayed.</td></tr>
<tr><td>Post Condition:</td><td colspan="3">System on/off or Space altered or Message displayed.</td></tr>
</table>

Activity Diagram

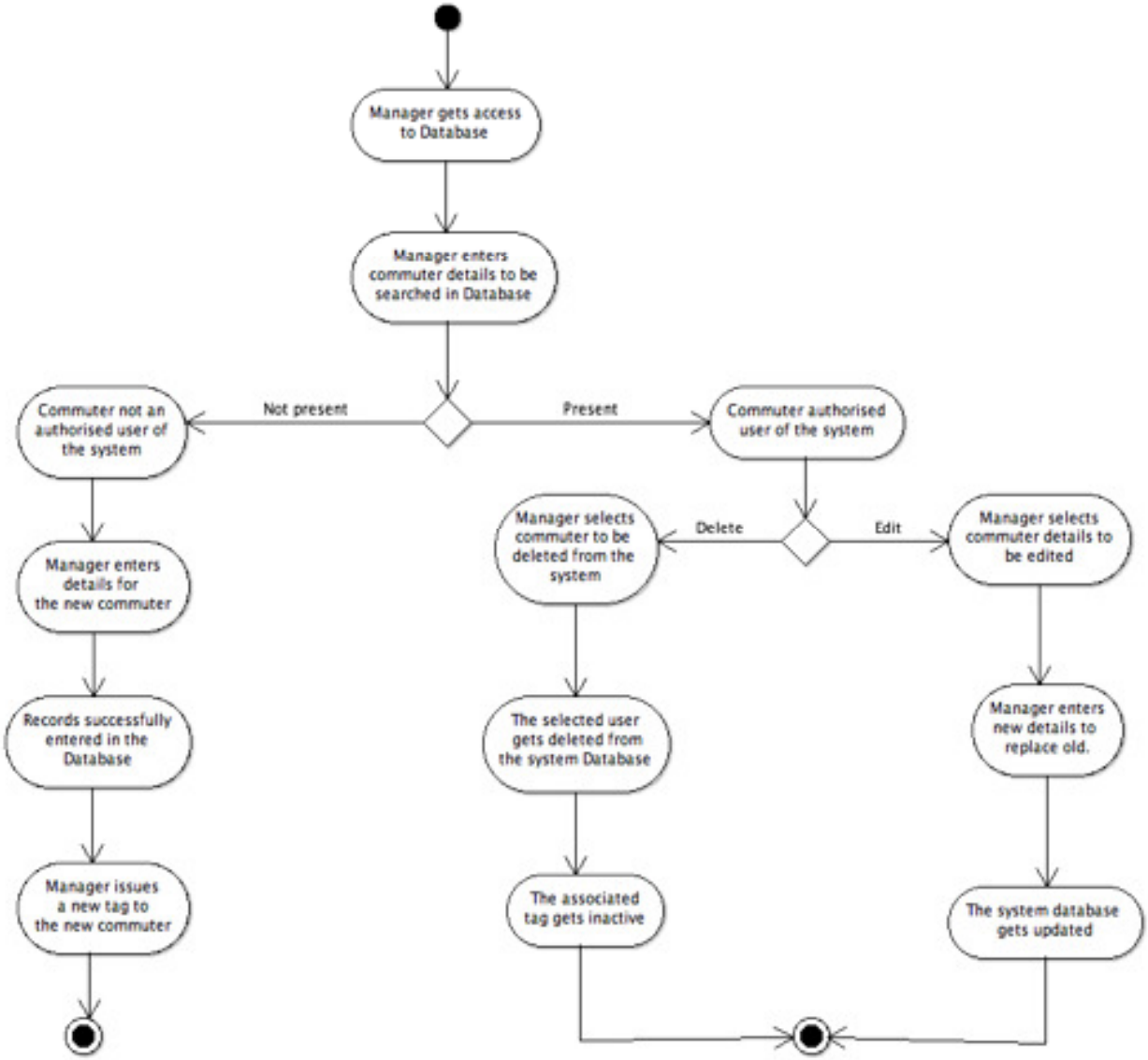


2.7 Update/New user database

<table>
<tr><td>Use-Case Name:</td><td colspan="3">Update/New member table</td></tr>
<tr><td>Primary Actor:</td><td colspan="3">Manager</td></tr>
<tr><td>Other Actors:</td><td colspan="3">None</td></tr>
<tr><td>Description:</td><td colspan="3">This use case describes the event of adding/removing/editing the access permission of a parking lot member.</td></tr>
<tr><td>Assumptions:</td><td colspan="3">System on.</td></tr>
<tr><td>Precondition:</td><td colspan="3">Manager authorized to access database.</td></tr>
<tr><td>Initiation/Trigger:</td><td colspan="3">Manager enters details to be searched.</td></tr>
<tr><td rowspan="6">Typical Course Of Events /Dialog:</td><td colspan="3">Actor Action</td></tr>
<tr><td colspan="3">Step 1: Manager gets access to the database.</td></tr>
<tr><td colspan="3">Step 2: Manager enters commuter details to search in the database.

Step 3: Database shows the result in the form of commuter details present/absent.</td></tr>
<tr><td>Step 4.a: Manager enters details for the new commuter in the database.</td><td colspan="2">Step 4.b: Manager selects the commuter details to delete/update.</td></tr>
<tr><td>Step 5a: Database is updated by the new details entered.</td><td>Step 5.b.1: The selected commuter gets deleted from the system database.</td><td>Step 5.b.2: Manager selects details to be updated for the commuter.</td></tr>
<tr><td>Step 6: Manager issues a new tag for the new commuter.</td><td>Step 6.b.1The associated tag gets deactivated.</td><td>Step 6.b.2: Details of the commuter gets updated in system database.</td></tr>
<tr><td>Conclusion:</td><td colspan="3">This use case concluded when system database is updated in the form of new- edited or deleted details of the commuter.</td></tr>
<tr><td>Post Condition:</td><td colspan="3">Database is updated by the action.</td></tr>
</table>

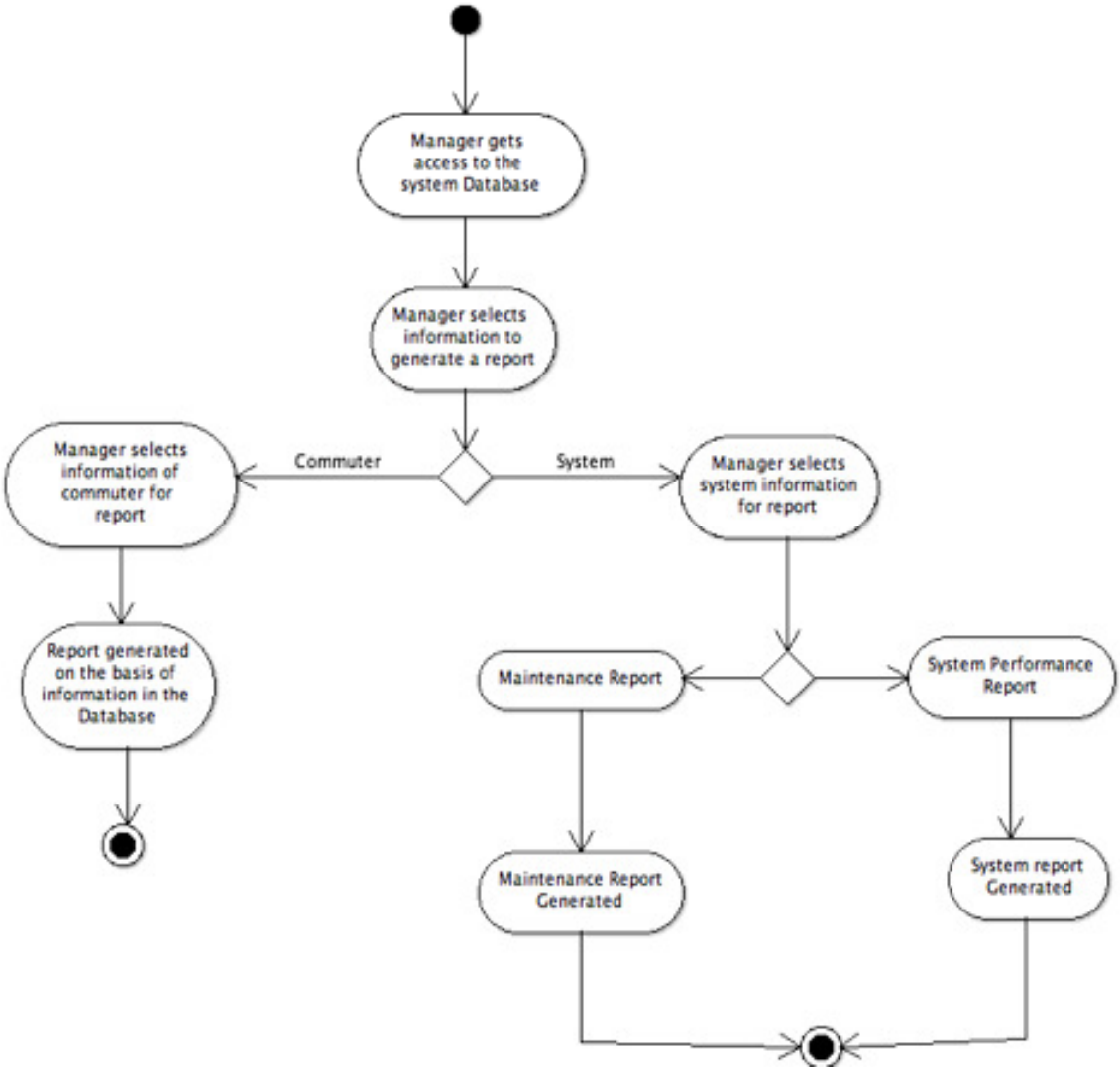
Activity Diagram



2.8 Generate Report

<table>
<tr><td>Use-Case Name:</td><td colspan="2">Generate Report</td><td></td></tr>
<tr><td>Primary Actor:</td><td colspan="3">Manager</td></tr>
<tr><td>Other Actors:</td><td colspan="3">None</td></tr>
<tr><td>Description:</td><td colspan="3">This use case describes the event of creating a report based on the information collected by the system.</td></tr>
<tr><td>Assumptions:</td><td colspan="3">System on.</td></tr>
<tr><td>Precondition:</td><td colspan="3">Manager authorized to access database.</td></tr>
<tr><td>Initiation/Trigger:</td><td colspan="3">Manager enters details to be searched.</td></tr>
<tr><td rowspan="6">Typical Course Of Events /Dialog:</td><td colspan="3">Actor Action</td></tr>
<tr><td colspan="3">Step 1: Manager gets access to the database.</td></tr>
<tr><td colspan="3">Step 2: Manager selects information to generate the report.</td></tr>
<tr><td>Step 3.a: Manager selects information of commuter for report.</td><td colspan="2">Step 3.b: Manager selects system information for report.</td></tr>
<tr><td>Step 4.a: Report generated on the basis of the information in the database.</td><td>Step 4.b.1: Manager opts for maintenance report.</td><td>Step 4.b.2: Manager opts for system performance report</td></tr>
<tr><td>Step 6: Manager issues a new tag for the new commuter.</td><td>Step 6.b.1: Maintenance report generated.</td><td>Step 6.b.2: Performance report generated</td></tr>
<tr><td>Conclusion:</td><td colspan="3">This use case concluded when system creates a report on the requested option by the manager.</td></tr>
<tr><td>Post Condition:</td><td colspan="3">Report from the database is generated.</td></tr>
</table>

Activity Diagram



3 REQUIREMENTS/ GOALS & SCENARIOS

.

3.1 Goals and Scenarios

The use case describes all that can happen in the achievement of the system objective. Use Cases drive the requirements engineering phase of development. The various Goals and Scenarios critical to the system are depicted below '

Goal 1: Operators must be able to edit and send messages to Display Boards

Scenario 1.1: Operator will send messages from the system to the display board.

Scenario 1.2: Operator wants to compose a new message that is not available in the list.

Goal 2: There must be an on/off button at the control center

Scenario 2.1: Emergency shut down required for maintenance.

Scenario 2.2: Control center to be evacuated in case of fire etc

Goal 3: Operators must be able to stop displaying current message on Display Boards.

Scenario 3.1: A higher priority message or warning could be waiting to be displayed.

Scenario 3.2: Displayed signs would no longer be valid.

Goal 4: The Display Board must be "all weather proof"

Scenario 4.1: Display board will be installed in outdoors as well as indoors.

Scenario 4.2: Display board will be installed in far south as well as far north

Scenario 4.3: There is likelihood of snow, rain; heavy winds etc (should be sturdy enough to withstand adverse conditions)

Goal 5: There must be an indicator showing the status of the active Display Board

Scenario 5.1: System sends a message to the display board but the display board was disconnected.

Scenario 5.2: Display board out of service for maintenance.

Scenario 5.3: Network or hardware failure on remote site.

Goal 6: The system must store all information about every activity performed at each workstation.

Scenario 6.1: Administrator wants to check the logs for specific information.

Scenario 6.2: Periodic reports to be generated to check system performance.

Goal 7: The system must be secure

Scenario 7.1: Unauthorized person tries to login.

Scenario 7.2: Operator tries to change system settings, which he is not authorized to.

Goal 8: The display board/sensor system must have backup power

Scenario 8.1: Power failure occurs.

Scenario 8.2: Power cable is broken due to accident.

Goal 9: The sensor must be capable of detecting any vehicle

Scenario 9.1: Vehicle might be small in dimensions.

Scenario 9.2: Vehicle might be parked wrong.

Goal 12: The system must be capable of reporting any system failure to the manager.

Scenario 12.1: The system is not displaying information.

Scenario 12.2: The sensor is unable to recognize vehicles parked.

3.2 Requirements:

1. The system must contain a database.

    1.1. The database should contain all the information about the commuters.

    1.2 The database must allow retrieval and addition of information.

2. The system must allow only the authorized users to enter the parking lot.

    2.1 The entrance gate control unit should open after the authorization of the commuter.

    2.2 The entrance sensor should verify the authorization through the database.

3. The system must display parking space availability.

    3.1 The system should display the number of occupied spaces.

    3.2 The system should display the number of vacant spaces.

    3.3 The system should display messages from the manager during special instances.

    3.4 The message should take less than 10 seconds from the control center to the display boards.

    3.5 The system should display message on the display board for 3 minutes unless and until stated otherwise.

4. The system must receive a confirmation message for every message sent on the display screen.

5. The system must have three parking spaces available.

    5.1 The system should have open parking spaces.

    5.2 The system should have handicap parking spaces.

    5.3 The system should security/service parking spaces.

6. The system should have individual display boards for all parking spaces.

    6.1 The display boards must display green light for available spaces.

6.2 The display boards must display red light for unavailable spaces.

6.2 The display boards must display blue light for handicap parking spaces.

7. The system must receive and send information to and from the unit controllers.

7.1 The unit controller must receive information from sensors.

7.2 The unit controller must display information on individual display boards.

7.3 The unit controller must send information to control system.

8. The system should have an indicator to indicate the status of display boards.

8.1 The system must have red color indicator for inactive display, green color indicator as active display board, and orange color to indicate a hardware failure of the display board.

8.2 Every display board must send feedback to the control center every 5 seconds to display it as alive.

9. The system should recognize the parked vehicle.

9.1 The system sensor must recognize the parked vehicle irrespective of its size.

9.2 The system sensor must recognize and report wrongly parked vehicle.

10. The system should allow for redefining the parking spaces.

10.1 The system must allow altering the parking spaces in case of maintenance.

11. The system should allow the user to take over during troubleshooting

11.1 The system must allow user to take over during system/sub-system failure.

11.2 The system must allow user to create/edit messages on main display board.

11.3 The system must allow user control while maintenance process.

12. The control center shall have a single button to turn off the system.

12.1 The button shall be a physical button present on the computer of the operator.

12.2 The button shall have an LED on its side indicating the status of the system.

12.3 The button shall have a clearly marked sign of system shut down.

13. The control center shall have security.

13.1 The control center shall have role based security system.

13.2 Role in the control center shall include supervisor, maintenance and operator.

13.3 The system shall have a counter for number of attempts.

3.3 Requirements Mapping:

SUBCOMPONENTS:

1. Parking Gate
2. Parking Space
3. Manager Interface
4. Display
5. Controller

UML DIAGRAMS

1. 2.3- Car Arriving
2. 2.4- Car Leaving
3. 2.5- Display Parking Space Availability
4. 2.6- System Privileges/Utilities
5. 2.7- Update/New user database
6. 2.8- Generate Report

REQUIREMENTS	ASSOCIATED COMPONENT	UML DIAGRAMS
REQ 1.0	Controller, Parking gate,	2.3, 2.7, 2.8
REQ 2.0	Controller, Parking gate	2.3, 2.7
REQ 3.0	Manager interface, Display	2.6
REQ 4.0	Controller, Display	2.5
REQ 5.0	Display, Parking space	2.3, 2.5,2.6
REQ 6.0	Parking space	2.3, 2.4
REQ 7.0	Controller, Parking space, Display	2.3, 2.5
REQ 8.0	Controller	2.3
REQ 9.0	Parking space	2.3, 2.5
REQ 10.0	Manager interface, Parking space	2.6
REQ 11.0	Manager interface, Display	2.6
REQ 12.0	Manager interface, Controller	2.6
REQ 13.0	Manager interface	2.6

4 DESIGN ANALYSIS & VALIDATION

The UML models are very helpful at hashing out all the different subcomponents of the system and their interaction. However, when creating the UML diagrams, the system designer may inadvertently create states that are never reached or legal transitions that if taken would put the system in a deadlock. These design mistakes are not usually identified and corrected until the software is being implemented or out on the field making the mistakes hard and costly to fix.

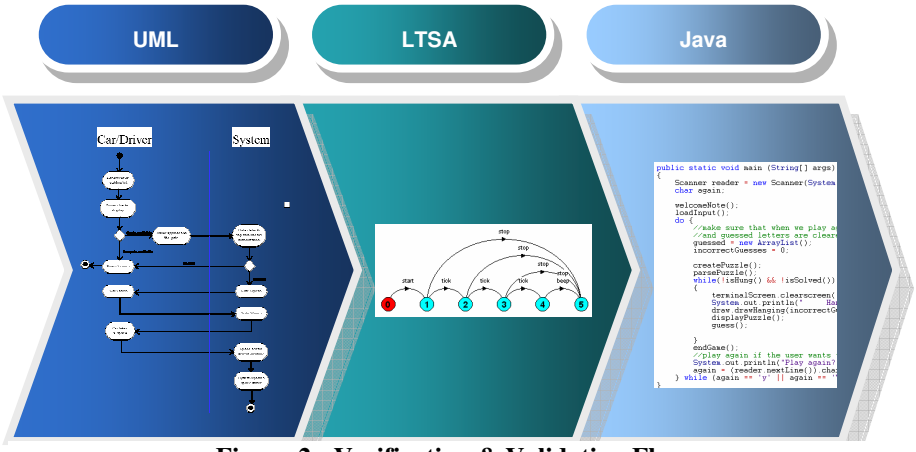


Figure 2 - Verification & Validation Flow

With that situation in mind, we decided to implement the design flow shown in Figure 2 - Verification & Validation Flow. First the UML was created as described in section 2. Based on the UML sequence diagram an LTSA model is created to check for potential deadlock or unreachable states. Once the parking lot controller design is complete, it was then implemented using Java to further verify its coherence and to validate the interaction with the external actors (e.g. car/driver).

The car arriving sequence diagram (Figure 3 - Car Arriving Sequence Diagram) was created using the Argo shareware. Using that as a starting point and making use of LTSA-MSC plug-in, we came up with the high level sequence of messages (Figure 4 - hSMC View) and the subcomponents interaction (Figure 5 - Trace View).

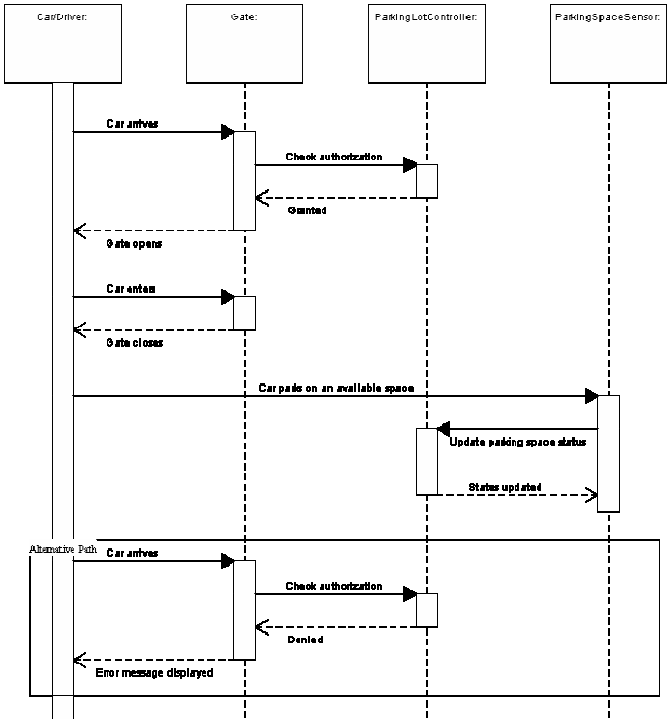


Figure 3 - Car Arriving Sequence Diagram

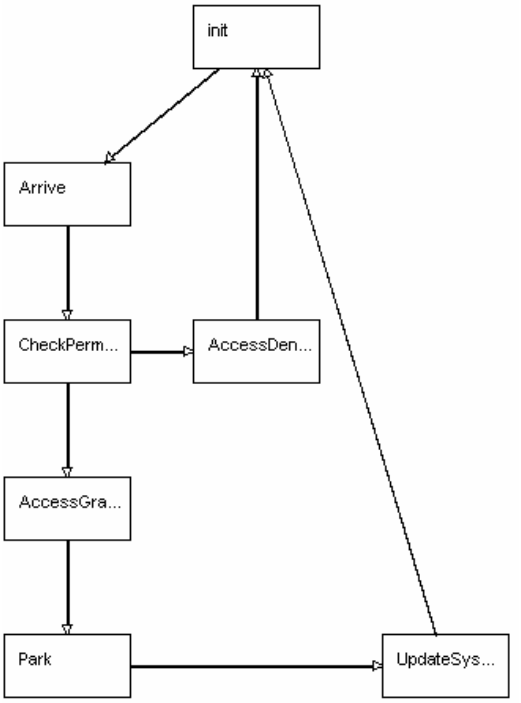


Figure 4 - hSMC View

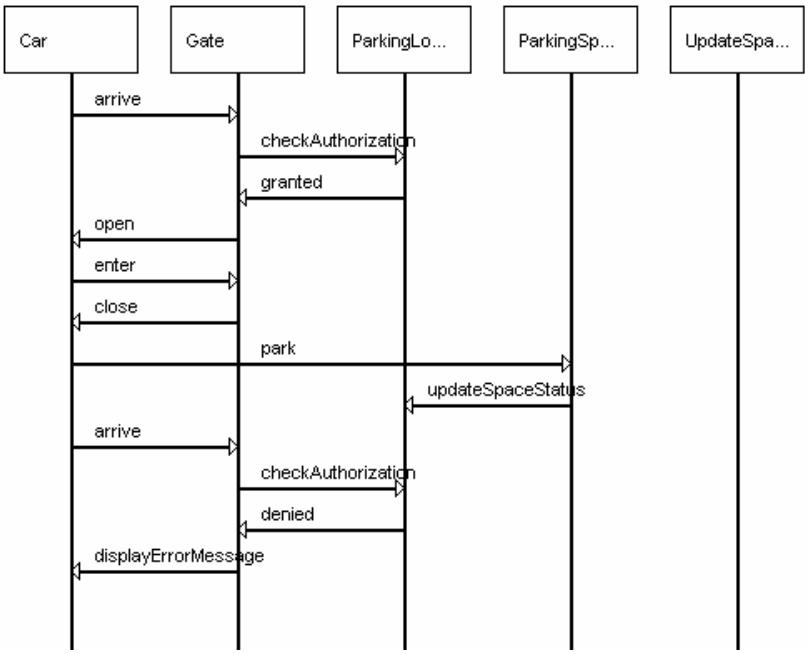


Figure 5 - Trace View

With the views above generated, the LTSA tool was able to create a textual notation of the system also known as FSP or Finite State Process. Once the FSP is created, we were able to check our model for safety and progress. The safety check looks for states with no outgoing transitions, which are potential deadlocks in the system. The progress check, also called liveness, this check verifies that whatever state a system is in, it is always the case that a specified action will eventually be executed. Both checks passed for our design. The resulting FSP model for subcomponents as well as the architecture is shown below.

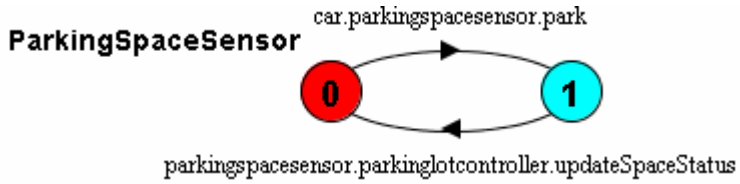


Figure 6 - Parking Space Sensor Model

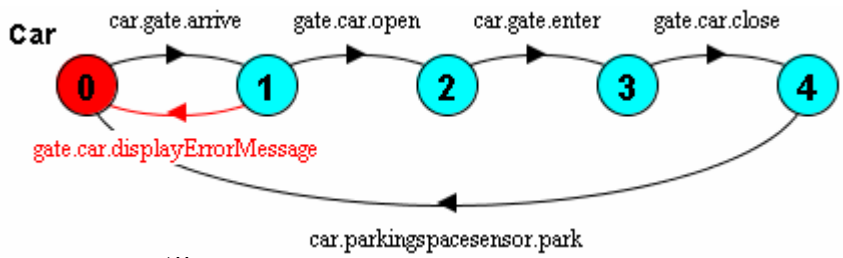


Figure 7 - Car Model

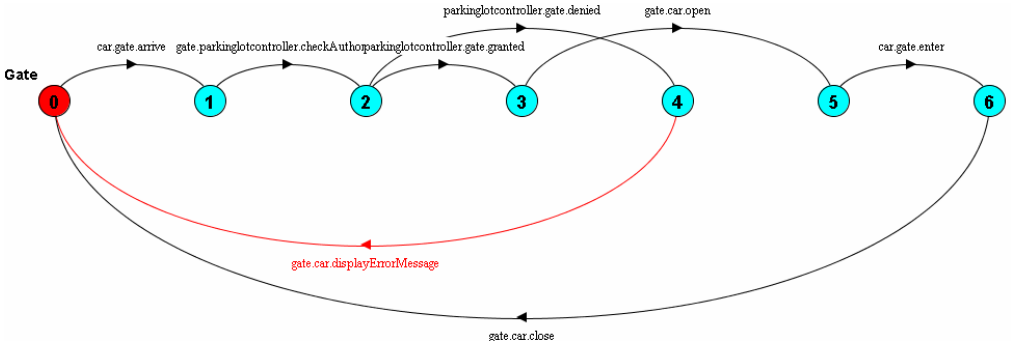


Figure 8 - Gate Model

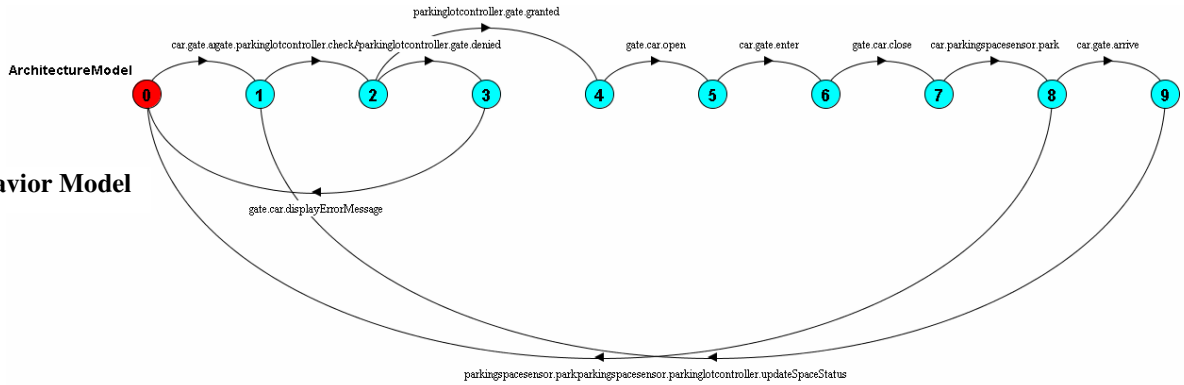


Figure 9 - System Behavior Model

The architecture model above was used to animate and check the behavior of the overall system. Figure 10 - Animation of Parking Lot Controller shows a captured image of the LTSA tool animator window. The list on the right presents all possible transitions. However, the green check mark indicates the valid transitions from the state the system is in. That way the model can be traversed from state to state to make sure the model behaves as expected. The animator was used to test a group of scenarios against the model and the outcome was verified.

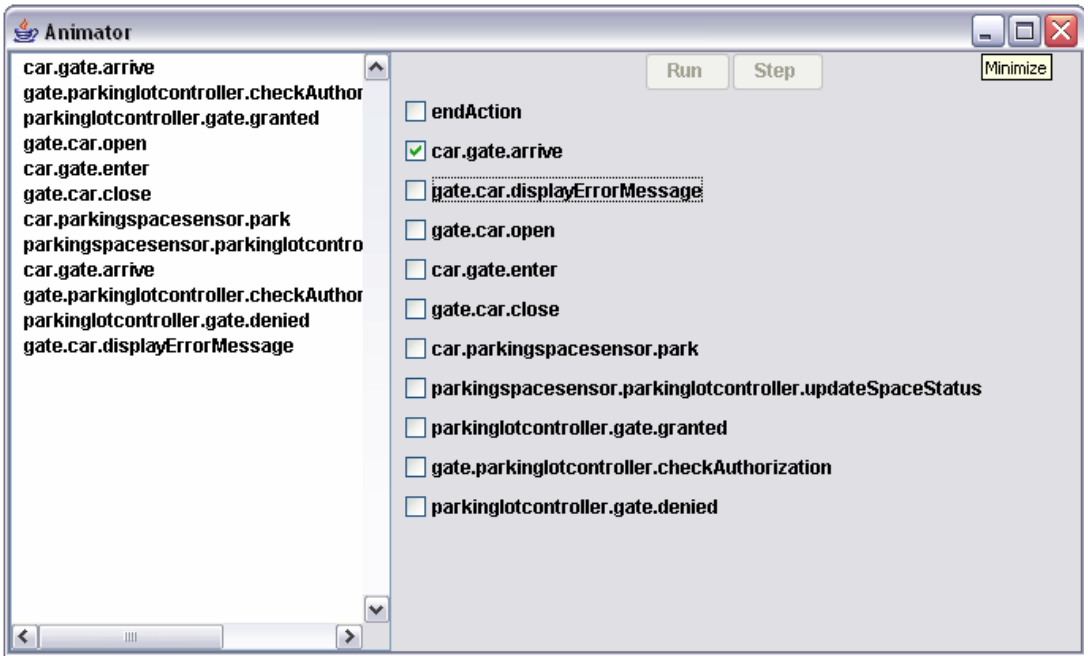


Figure 10 - Animation of Parking Lot Controller

The last step in our flow was to validate the interaction between the parking lot controller and the actors external to the system. To do that, a Java applet was created where classes were created to emulate each actor and the parking lot controller was implemented in a class of itself. The screenshot, Figure 11 - Parking Lot Java Applet, is of the resulting applet.

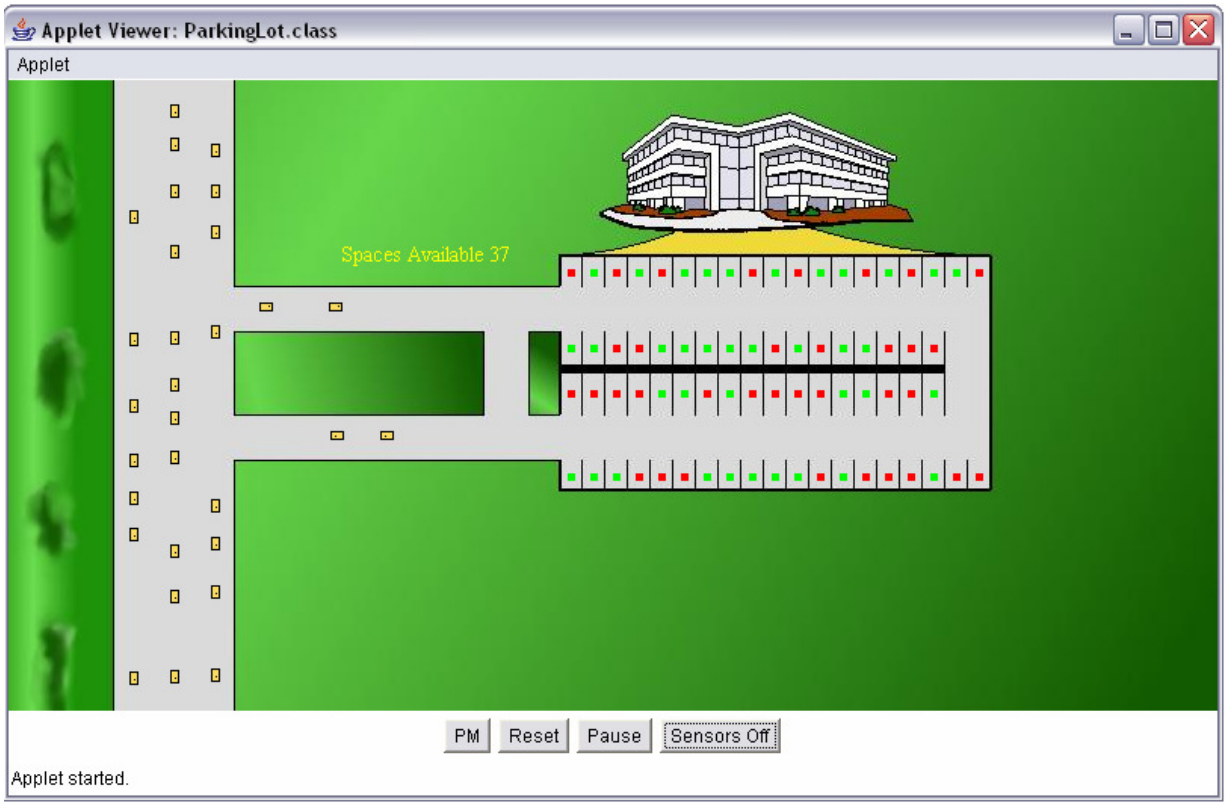


Figure 11 - Parking Lot Java Applet

5 CONCLUSION

For this project, the goal was to take the idea of an occupancy aware parking lot through the design process as far as possible given the time and resources. At first the system details were hashed out through the use of use case scenarios. These scenarios were then further refined and modeled using UML diagrams. These first steps are widely applied during the design process of systems. For this project however, we decided to validate the UML models using a well defined and repeatable process rather than the usual model review meetings with team members approach. The method used was based on composing joint sets of transition state machines using the FSP language and then check for unexpected behavior using a set of tools.

Due to time constraints, we decided to go through the validation and verification flow with the Car Arriving use case only. The exercise was successful and we were able to actually implement part of the system at a higher level, using Java, to simulate its behavior. The available tools out there, such as LTSA, are very helpful at discovering hidden potential paths within the models and verifying that they work as intended. However, it is best suited for small size systems at the present moment and need to be further developed before mainstream adoption is possible. One important feature would be the support for creating a library of low level models that could be reused over and over as building blocks for creating models of larger systems.