Software Engineering
Semester Project
Team Dory

Monmouth University
Software Engineering 104

Courtney A. Locke

Summer 2013

Table of Contents

Scribbler S2 Project

Problem Statement

In this project, we will demonstrate how a robot can come off cute, or uncute, to a person, based off movement and sounds alone.

Introduction and Overview

This project was developed in reaction to a previous project, shown in an exhibit from New York City's Museum of Modern Art, named the Tweenbot project. This project had made use of small robots, which were placed in various locations in NYC. The missions of the robots were to make it from their placed location, to the MOMA. During the various robots journeys, they would become "stuck", although they still made it to the MOMA due to the fact that people would help them by placing them in the right direction of the MOMA. This led to the conclusion that people are so helpful even to the extent of aiding a robot on its journey.

This is where the Scribble S2 Project comes in. We hypothesized that the real reason people helped these small robots, were due to the fact that they were cute, non-threatening, and helpless-looking robots. We want to figure whether or not that hypothesis is true.

In order to test our theory, we will program a plain robot, specifically a Scribbler S2, in two modes. One mode being programmed to be cute, and another mode to be programmed as uncute. Then, the robots will be released into society to see how they react with the robots in each mode. Our robots will also be able to react to bumping into an obstacle.

Process Model

For this project, I will be using the Waterfall Process Model, but the modified version. I chose this due to the fact that I am giving a one time delivery of our project into society, so it seemed the best fit, but the modified version instead of the strict waterfall model, since there will be much back and forth between coding and testing this project.

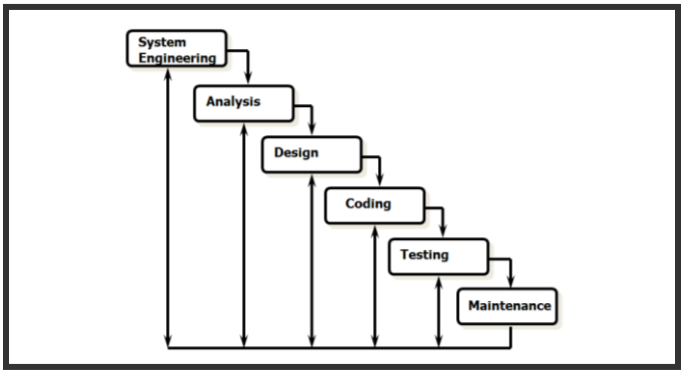


Figure 1, *Modified Waterfall Model***.**

From start to end, this projects process with need to include all of the basic elements as seen in Figure 1. Referring to the diagram below, one can see what will be included in each stage of the model.

Stage	What must be done/completed before going to next stage of model
System Engineering	Deciding what requirements are needed in the production of this project
Analysis	Analyzing what must be done in each mode to give off the desired cute or uncute trait
Design	Designing what will be included in each separate mode
Coding*	Coding of both modes
Testing*	Testing of both modes to ensure the cute and uncute emotion is clear
Maintenance	Fix any final issues
*Due to the use of the modified waterfall model, project will spiral through these multiple times	

As I progress through the Modified Waterfall Model, each step once complete, in theory, should not be able to be gone back to. This would be though if I were using the Strict Waterfall Process Model. In using the modified waterfall model, as I continue through the process, I can go back if needed. This does not eliminate the need for conditions to go from one stage to the next though.

If the basic needs of the stage are not completed, one cannot continue onto the stage and expect a great product. For instance, I cannot go onto the Analysis stage without gathering all the requirements my project will need, due to the fact that it will cause problems later down the line in my project. I cannot go onto the Design stage if I have not completed the Analysis Stage, since the analysis stage gives me the information needed for design. I cannot go onto the Coding stage without completing the design because I won't know what I am supposed to code. I cannot then go to the Testing stage because there will not be proper code to test, and finally, I cannot go onto the Maintenance Stage because the full project is not completed therefore Maintenance cannot really be done. Ensuring each stage is completed before moving on creates an easy flow and momentum of the project.

System/Software Requirements

Use Cases

Use Case #		**Robot must act in either a cute or uncute way**
Description		User must be able to activate either a cute, or uncute mode of performance. This is done by coordination of finger placement over the light sensor area of the robot.
Actors		User
Steps	1.	Place robot on floor
	2.	Turn on robot
	3.	Using fingers, cover corresponding light sensor area
	4.	Observe human response to displayed mode

Use Case #		**Robot crossing a large black line on an otherwise white surface**
Description		The robot must be able to detect a thick black line on an otherwise white surface using its sensors
Actors		Robot
Steps	1.	Place robot on floor
	2.	Turn on robot
	3.	Using fingers, cover corresponding light sensor area
	4.	Let robot “behave” in corresponding mode
	5.	Observe robot reaction when reacting to crossing a large black line on an otherwise white surface

Use Case #		**Robot reacting to bumping into an obstacle**
Description		The robot must react to bumping into an object and react
Actors		Robot
Steps	1.	Place robot on floor
	2.	Turn on robot
	3.	Using fingers, cover corresponding light sensor area
	4.	Let robot “behave” in corresponding mode
	5.	Observe robot reaction when reacting to bumping into an object

Requirements Description

Req # 1	**Category:** Functional	**Use Case #:** 1	**Priority:**
Description: Robot must have a cute mode		**Source:** Project Description	
Criterion: Robot must be programed to have a mode giving off a “cute” appearance, being limited to only its movements, lights, and sounds.			
Dependencies: User must put in corresponding mode, Light Sensors		**Conflicts:** Put in wrong mode	

Req # 2	**Category: Functional**	**Use Case #:** 1	**Priority:**
Description: Robot must have an uncute mode		**Source:** Project Description	
Criterion: Robot must be programed to have a mode giving off an “uncute” appearance, being limited to only its movements, lights, and sounds.			
Dependencies: User must put in corresponding mode, Light Sensors		**Conflicts:** Put in wrong mode	

Req # 3	**Category: Functional**	**Use Case #: 2**	**Priority:**
Description: Robot must be able to detect a thick black line on an otherwise white surface		**Source:** Project Description	
Criterion: Robot must be able to detect a thick black line on an otherwise white surface, and react.			
Dependencies: Line, Sensors		**Conflicts:** Cannot see line with sensors	

Req # 4	**Category:** Functional	**Use Case #:** 2	**Priority:**
Description: Robot must be able to follow a thick black line on an otherwise white surface		**Source:** Project Description	
Criterion: Robot must be able to follow a thick black line on an otherwise white surface, and if loses the line, attempt to find it again and follow again			
Dependencies: Line, Sensors		**Conflicts:** Cannot see line with sensors	

Req # 5	**Category:** Functional	**Use Case #:** 3	**Priority:**
Description: Robot must be able to move in multiple directions		**Source:** Project Description	
Criterion: Robot must be able to move in a forward and backwards motion and rotate using two wheels			
Dependencies: Wheels		**Conflicts:** Getting stuck	

Req # 6	**Category:** Functional	**Use Case #:** 2,3	**Priority:**
Description: Robot must be able to utilize its sensors		**Source:** Use case 2, 3	
Criterion: Robot must be able to utilize its sensors to sense an obstacle, line			
Dependencies: Wheels, Lines, Obstacles		**Conflicts:** No sensors working	

Req # 7	**Category:** Functional	**Use Case #:** 1	**Priority:**
Description: Robot must utilize its lights		**Source:** Requirement 1, 2	
Criterion: Robot must utilize light feature to aid in giving cute or uncute effect			
Dependencies: Functioning lights, mode		**Conflicts:** Non-functioning lights	

Req # 8	**Category:** Functioning	**Use Case #:** 1	**Priority:**
Description: Robot must utilize sound feature		**Source:** Requirement 1, 2	
Criterion: Robot must utilize sound feature to aid in giving cute or uncute effect			
Dependencies: Functioning sound, mode		**Conflicts:** Non-functioning sound system	

Req # 9	**Category:** Functioning	**Use Case #:** 1	**Priority:**
Description: Robots movements must be streamline		**Source:** Project Description	
Criterion: Robots movements must be streamline and fluid as if one, in order to appear natural			
Dependencies: Wheels, Programmer		**Conflicts:** Broken code, User	

Req # 10	**Category:** Non-Functioning	**Use Case #:** 1	**Priority:**
Description: Robot must gain attention from humans		**Source:** Project Description	
Criterion: Robot must gain attention from humans in surrounding area in order to demonstrate its cute or uncute mannerism			
Dependencies: Human crowd		**Conflicts:** Absence **of humans**	

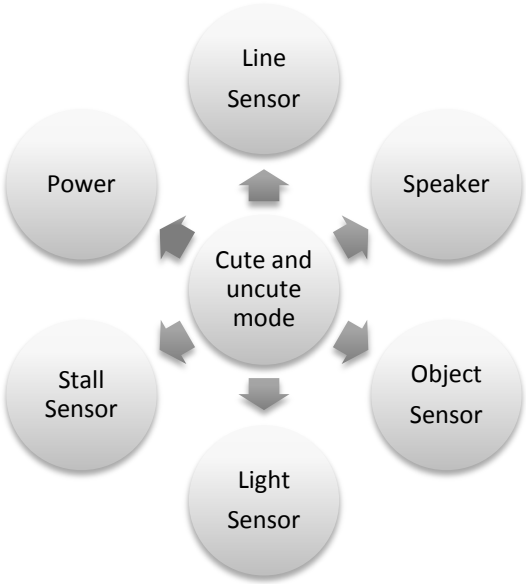
Software Design

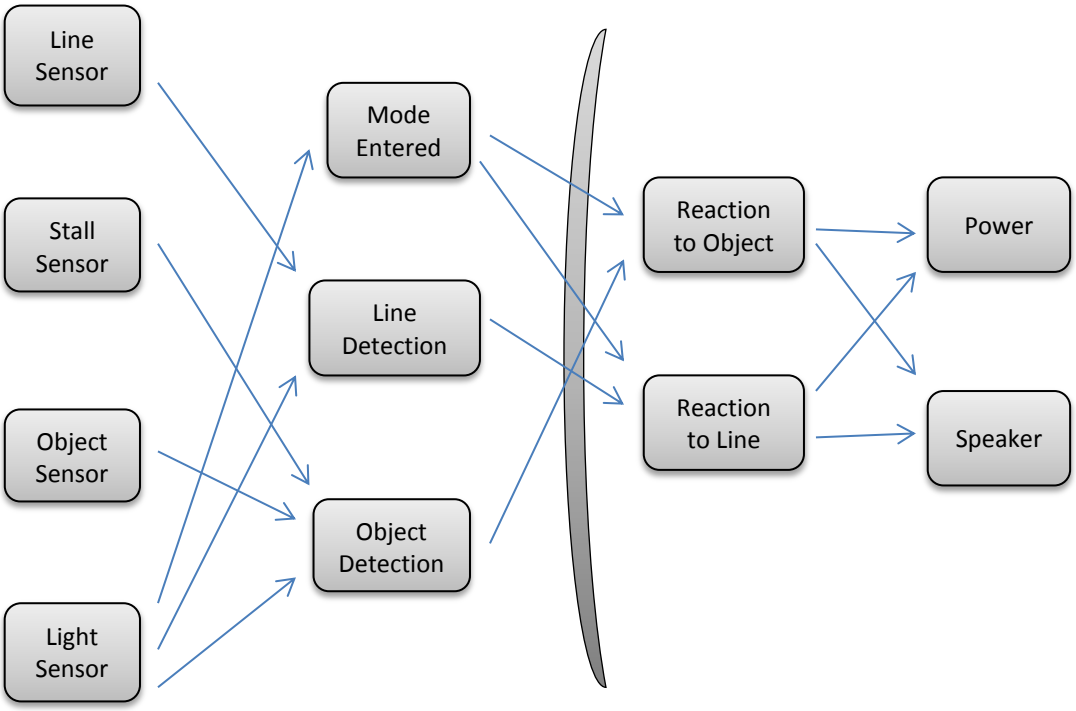
The Scribbler S2 robot will be programmed using a simple “drag and drop” interface by each student individually. The program created will have two modes, and two logical issues to compute. The two modes to be programmed are a cute, and uncute mode. The logical issues are bumping into an object, and crossing a line.

To enter into each separate mode, the student will have to program the robot to read a combination of light sensor coverings, corresponding with each program. Once the robot has recognized which program it is in, the robot will then perform the program. Within each mode, the robot will have to troubleshoot the interaction with an obstacle and a line, each pre-programmed into each mode how to solve each issue.

Through the programming of the software, each requirement mentioned earlier in the document will be fulfilled.

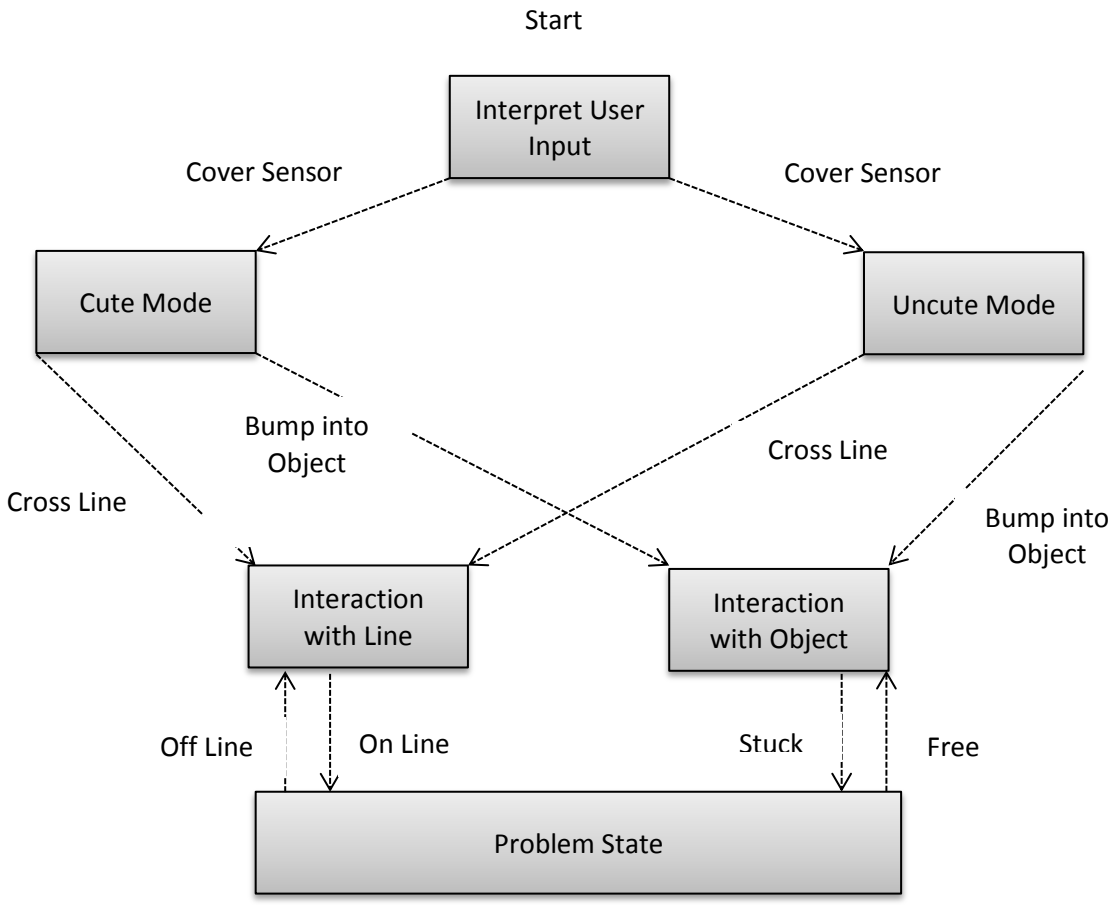
Data Flow Diagram





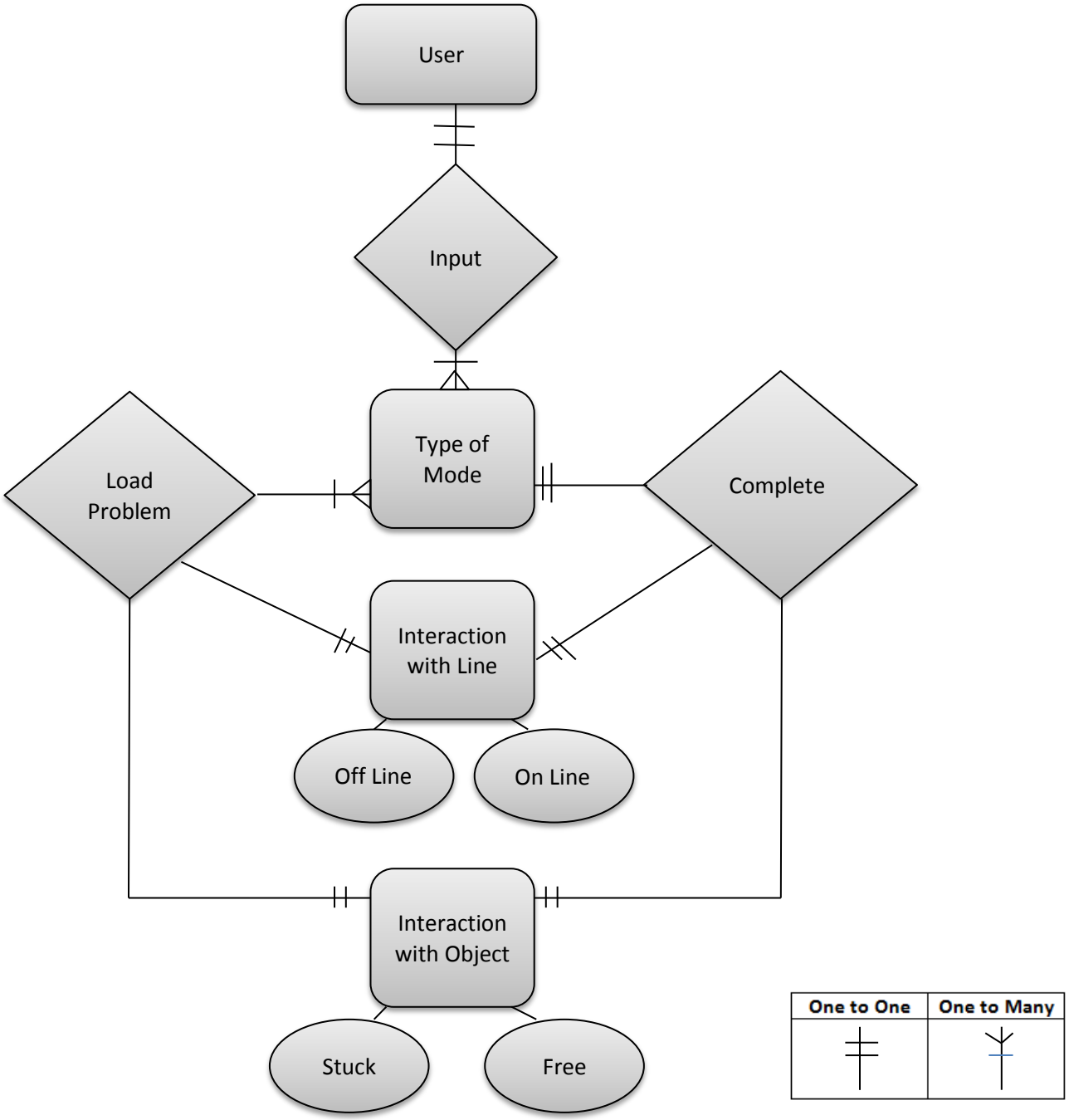
The diagram above is a visual representation of how information flows from the main processing unit, through the entirety of the program.

State Transition Diagram



The diagram above visually explains the states the robot will be in during the duration of the program.

Entity – Relationship Diagram



The diagram above visually explains the relationship between parts of the program.

Test Specification

Test Case # 1	**Who does Testing:** User	
Description: Robot must perform in cute/uncute mode determined by light sensor reading		**Setup:** Robot in hand or on surface and turned on
Criterion: Robot must be able to read light sensor input correctly, and then follow programming loop designated to specific input		
Dependencies: User, Light Sensors		**Expected Results:** Robot performs correct mode based on light sensor reading

Test Case # 2	**Who does Testing:** User	
Description: Robot must play introduction music corresponding to cute or uncute mode		**Setup:** Robot on surface
Criterion: Robot must be able to play music		
Dependencies: Speaker		**Expected Results:** Robot plays correct introduction music

Test Case # 3	**Who does Testing:** User	
Description: Robot detects and moves away from object on its left side		**Setup:** Robot on surface with obstacle in way on left side
Criterion: Robot must be able to use its sensors to avoid obstacles on its left side		
Dependencies: Sensors, Obstacle, LEDs, Speaker		**Expected Results:** Robot correctly avoids object, while left most side LED illuminates green, and plays a sound

Test Case # 4	**Who does Testing:** User	
Description: Robot detects and moves away from object on its right side		**Setup:** Robot on surface with obstacle in way on right side
Criterion: Robot must be able to use its sensors to avoid obstacles on its right side		
Dependencies: Sensors, Obstacle, LEDs, Speaker		**Expected Results:** Robot correctly avoids object, while right most side LED illuminates green, and plays a sound

Test Case # 5	**Who does Testing:** User	
Description: Robot detects and moves away from object directly in front		**Setup:** Robot on surface with obstacle directly in front
Criterion: Robot must be able to use its sensors to avoid obstacles directly in front of it		
Dependencies: Sensors, Obstacle, LEDs, Speaker		**Expected Results:** Robot correctly avoids object, while all three LEDs illuminate green, and plays a sound

Test Case # 6	**Who does Testing:** User	
Description: Robot detects and reacts to bumping into an obstacle		**Setup:** Robot on surface
Criterion: Robot must be able to use its stall sensor to react to bumping into object		
Dependencies: Sensors, Obstacle, LEDs, Speaker		**Expected Results:** Robot correctly avoids object, while middle LED illuminates green, and plays a sound

Test Case # 7	**Who does Testing:** User	
Description: Robot detects black line on its right side		**Setup:** Robot on surface and seeing black line on its right
Criterion: Robot must be able to detect a black line on an otherwise white surface and react		
Dependencies: Sensors, Black Line, Speaker		**Expected Results:** Robot correctly turns toward line, plays music

Test Case # 8	**Who does Testing:** User	
Description: Robot detects black line on its left side		**Setup:** Robot on surface and seeing black line on its left
Criterion: Robot must be able to detect a black line on an otherwise white surface and react		
Dependencies: Sensors, Black Line, Speaker		**Expected Results:** Robot correctly turns toward line, plays music

Test Case # 9	**Who does Testing:** User	
Description: Robot detects that it is on a black line		**Setup:** Robot on surface and seeing black line
Criterion: Robot must be able to detect a black line on an otherwise white surface and react		
Dependencies: Sensors, Black Line, Speaker		**Expected Results:** Robot reacts, plays music

Test Case # 10	**Who does Testing:** User	
Description: Robot gives off a cute performance		**Setup:** Robot on surface
Criterion: Robot must be able to give off a cute performance based solely on its movements and sounds		
Dependencies: Sensors, Motor, Speaker		**Expected Results:** Robot seems cute

Test Case # 11	**Who does Testing:** User	
Description: Robot gives off an uncute performance		**Setup:** Robot on surface
Criterion: Robot must be able to give off an uncute performance based solely on its movements and sounds		
Dependencies: Sensors, Motor, Speaker		**Expected Results:** Robot seems uncute

Test Case # 12	**Who does Testing:** User	
Description: Robot must move forward when on white surface		**Setup:** Robot on surface
Criterion: Robot must move in a forward motion when on a white surface without obstacles in view		
Dependencies: Sensors, Motor		**Expected Results:** Robot moves in straight line forward

Test Case # 13	**Who does Testing:** User	
Description: Robot is lifted off floor and should react		**Setup:** Robot in air
Criterion: Robot must react to being lifted up off the surface. Robot should continue as if on black line		
Dependencies: Sensors, Motor		**Expected Results:** Robot does the same as in Test Case 9

Traceability Table

Requirement #	Req. Description	Source of Req	SW Module(s) satisfying the Req	Test Case(s) testing the requirement
1	Robot must have a cute mode	Project Description	Computation	Test if in cute mode
2	Robot must have an uncute mode	Project Description	Computation	Test if in uncute mode
3	Robot must be able to detect a line	Project Description	Link	Test line detection
4	Robot must be able to follow a line	Project Description	Link	Test line detection
5	Robot must be able to move in multiple directions	Project Description	Link	Test for movement
6	Robot must be able to utilize its sensors	Use case 2, 3	Controller	Test for working sensors
7	Robot must be able to utilize its lights	Requirement 1, 2	Controller	Test if lights working
8	Robot must be able to utilize its sound feature	Requirement 1, 2	Controller	Test for sound
9	Robot movements must be streamline	Project Description	Controller	Test for streamline movement
10	Robot must gain attention from humans	Project Description		Observe Reactions from each mode

Conclusions

This project was very intriguing to me. Figuring out a way for a robot to come off cute or uncute was a challenging, but enriching experience. I felt testing and retesting software is extremely important to produce a high quality product, as well as to ensure that the robot came off in the proper manner, dependent on sensors.

Determining what was cute and uncute was a challenge in itself. Focusing on my cute mode, I wanted it to grab the viewer's attention based off sounds and movements, as though to be a performance. With the limited preinstalled sounds, I was forced to create my own sounds, inputted from basic piano sheet music. I also made my robot react to the objects and lines in a theatric way, such as rotating to music and moving different speeds. For my uncute mode, I utilized not so cute sounds, such as a frog sound and basic robot noises, as well as the same speed throughout all movements, in order to just make my robot seem "plain jane" and uncute compared to my vibrant musical cute mode. Unfortunately, sounds played the biggest part in giving off the modes sense of cute or uncute, although I attempted to show modes with movements too.

I also learned that in order to create a streamline program with the drag and drop s2 software, the sequences of each if else would have to be minimal. For example, in my cute mode, my robot would sometimes not react to an object, due to the fact that its previous action takes five seconds to complete, and it ran into an object three seconds in. Using spin language probably would have fixed this issue, but I unfortunately could not figure it out. Making the s2 program just very plain, meaning no cute or uncute attributes, made the program run extremely streamline, but our project required us to have those attributes, so the streamlined movement and accuracy was impacted a bit.

Through much trial and error, both in the programming of the S2 and the project report, I learned what is necessary to develop a strong product, both on paper and in function.