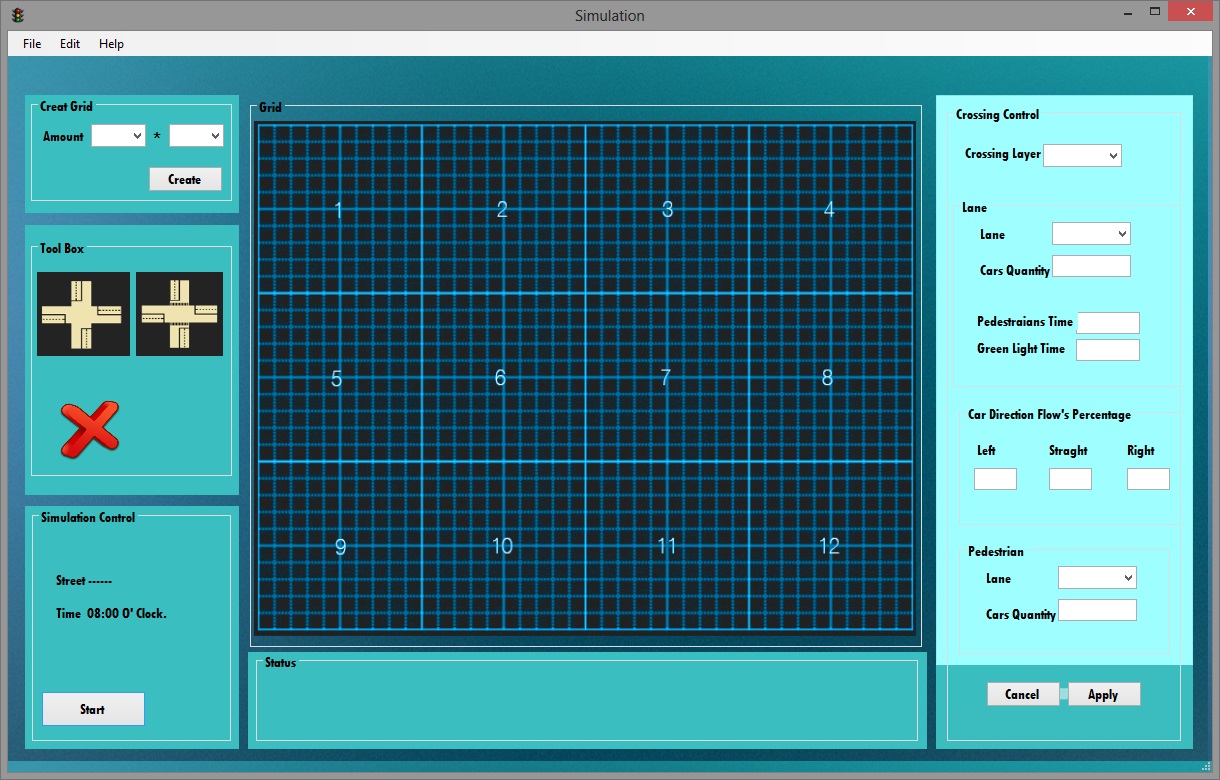
|  |  |
| --- | --- |
| DeSIGN DOCUMENT  Mikaeil Shaghelani Lor - Mervin Vrolijk - Bolarinwa Iruemiobe - Nibras Shawy - Tarwiya Al Ismaili | group C  Class diagram-Description of the classes and their members along with Some sequence diagrams.  **[ProCP- Traffic Light simulator ]**  14/OCT/2015 V0.0.1 |

1. **Table of Contents**
2. Table of content-------------------------------------------------------------1
3. Introduction------------------------------------------------------------------2
4. Class Diagrams--------------------------------------------------------------3
5. Description of classes----------------------------------------------------4-5
6. Interactions /Some sequence diagrams--------------------------------6-9
7. Assumptions and Risks------------------------------------------------9-10
8. **Introduction**



**Traffic Light Simulations**

Purpose

This design document describes the architecture and system design of a

Traffic simulator along with elaborating the entities, attributes

and their relations in respect to how they could be used to develop this

program most efficiently in order to achieve the decent simulator that can simulate

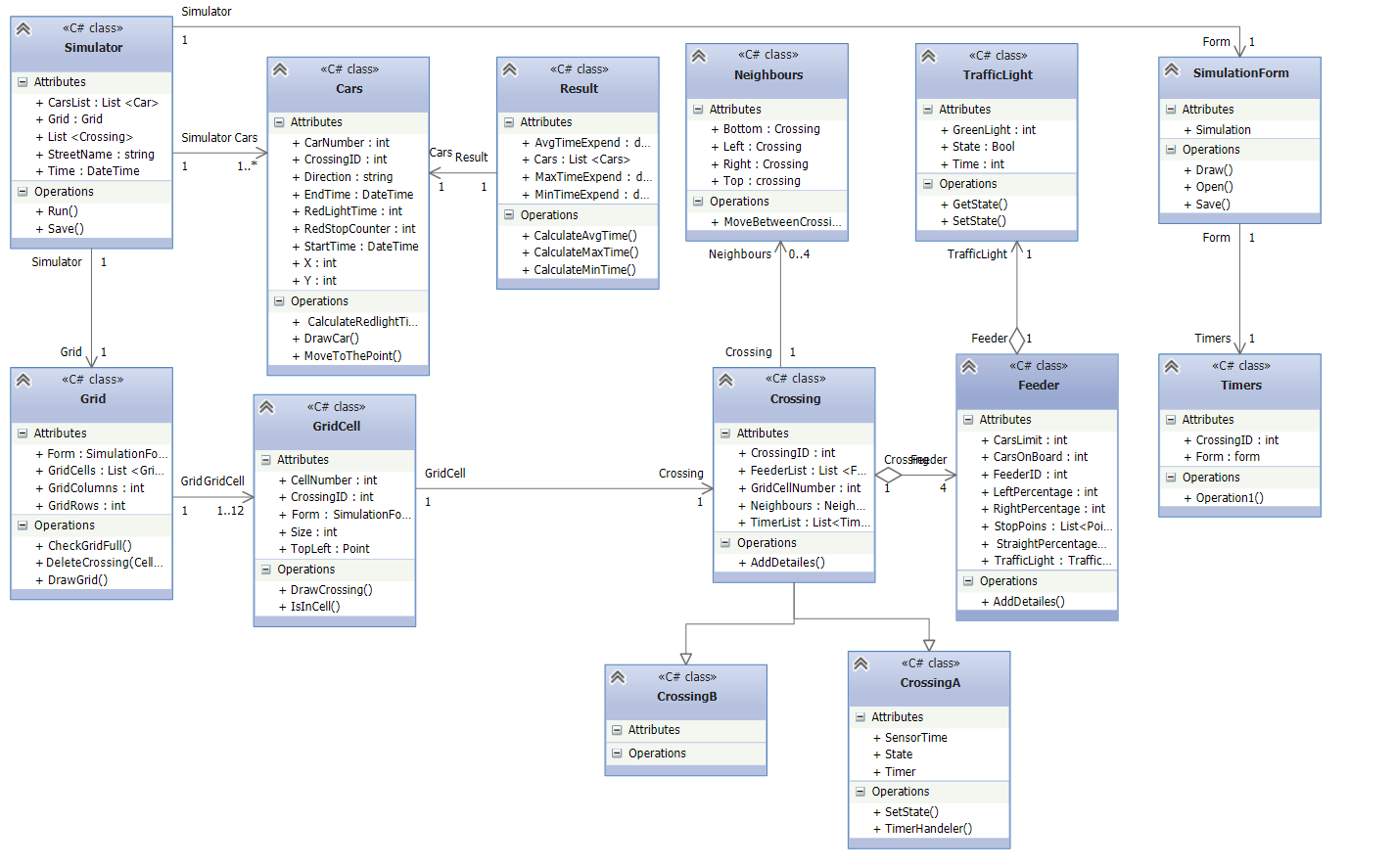
every possibilities.

1. Scope

The objective and the goal of this product is to simulate diffrent traffic flow situations to tests and find out the best usage of the traffic lights in order to decrease the number of accident in the town.

In this design document we provide description of our high level entities among with some design visualizations such as **class diagrams** and few of the most important **sequence diagram**s ,which would explain the interactions for some specific set of actions.

1. **Class diagram**

****

***Figure 3.1***

1. **Description of the classes and their members**

**Classes:**

* Simulation
* Grid
* Crossing
* GridCell

**/// Class Members description see figure 3.1**

All important elements that share the same characteristics will inherit from the element class (interface) such as AND Gate, NOT Gate, OR Gate.

LED and charge which will be discussed furthermore.

-Element

Inside the element class the state of elements will be stored which is independent from the output wire.

* get\_output\_state method will return the output state of the element independent if there's a wire or not and
* set\_output\_state(int new\_output\_state) method sets the output state of the element that is determined by the inputs that the element has received. In this case it is set without any influence from the inputs.
* output\_wire is the wire that will hold the output of the element.
* Wire[] input is an array of wires that will hold the input of the element.
* The number of input at a time will be set by set\_inputs(ref Wire[] wires) method.
* clear\_wire(Wire wire) Removes an input from the element.
* get\_input\_pin(uint pin) gets the pin and returns the wire at the selected pin. Position is an array of size two which will hold the positions of the element on the sketch.

-Image

Contains the image that the element will draw on the screen.

The space is the element that will occupy on the screen. It's needed for detecting a click on an element and choosing the input/output.

Every element will update the wires connected to it. The output wire will change according to what the input wires states are at the moment.

* Inside the nullshoot() method the possibility of an element to be updated will be checked.

There are some output wires, so if one of the inputs is cancelled, set the output to 0 to indicate that there is no current anymore.

Beside that in the charge class the source of the circuit will be calculated.

-Sketch class

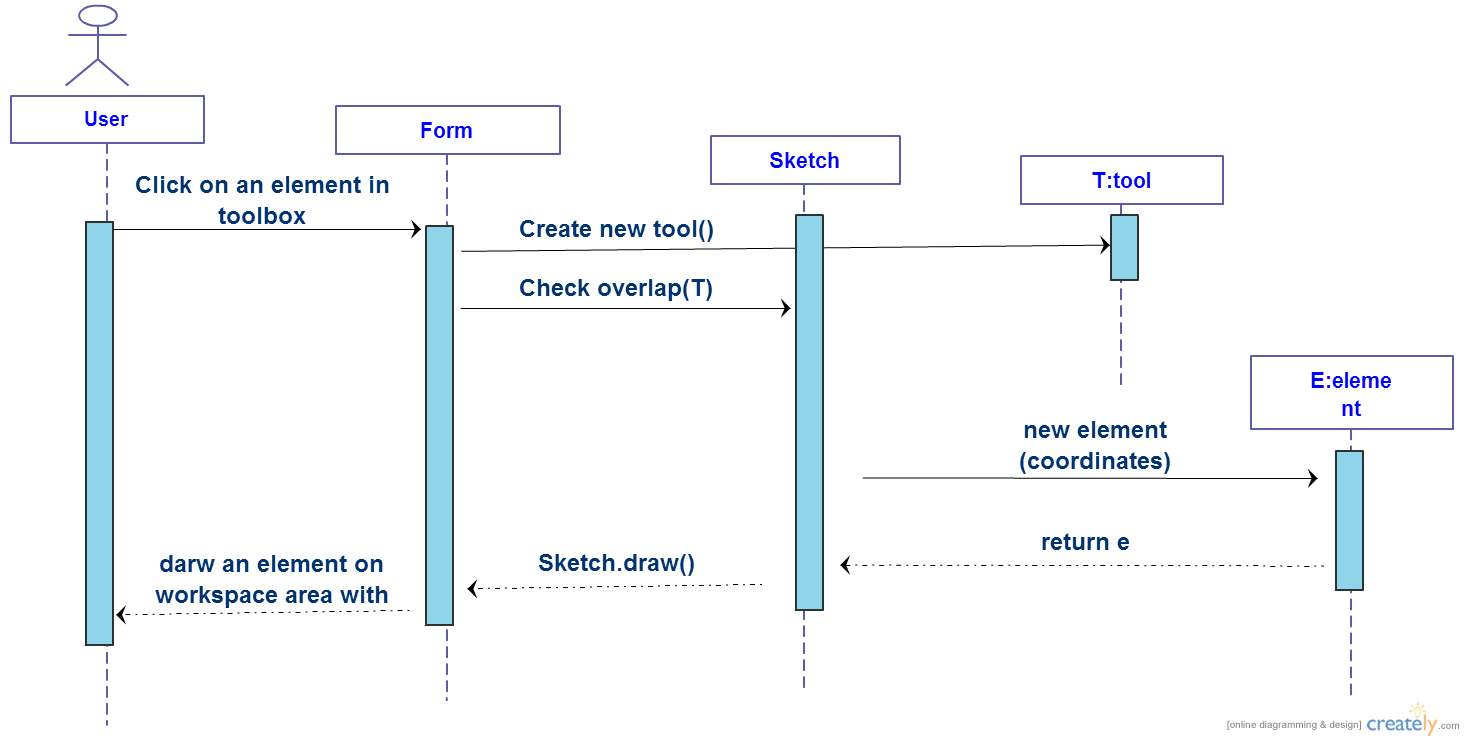
Sketch class is a controller in the form.it contains the details of the sketch such as name, size and list of wires and elements in the sketch, all the details of sketch can be set by this class.

Saving the file and image file will be done inside this class.

* The CheckOverlap(Element first, Element second) method ,inside the sketch class if the two elements are overlapping will be checked.

1. **Interactions/sequence diagrams**

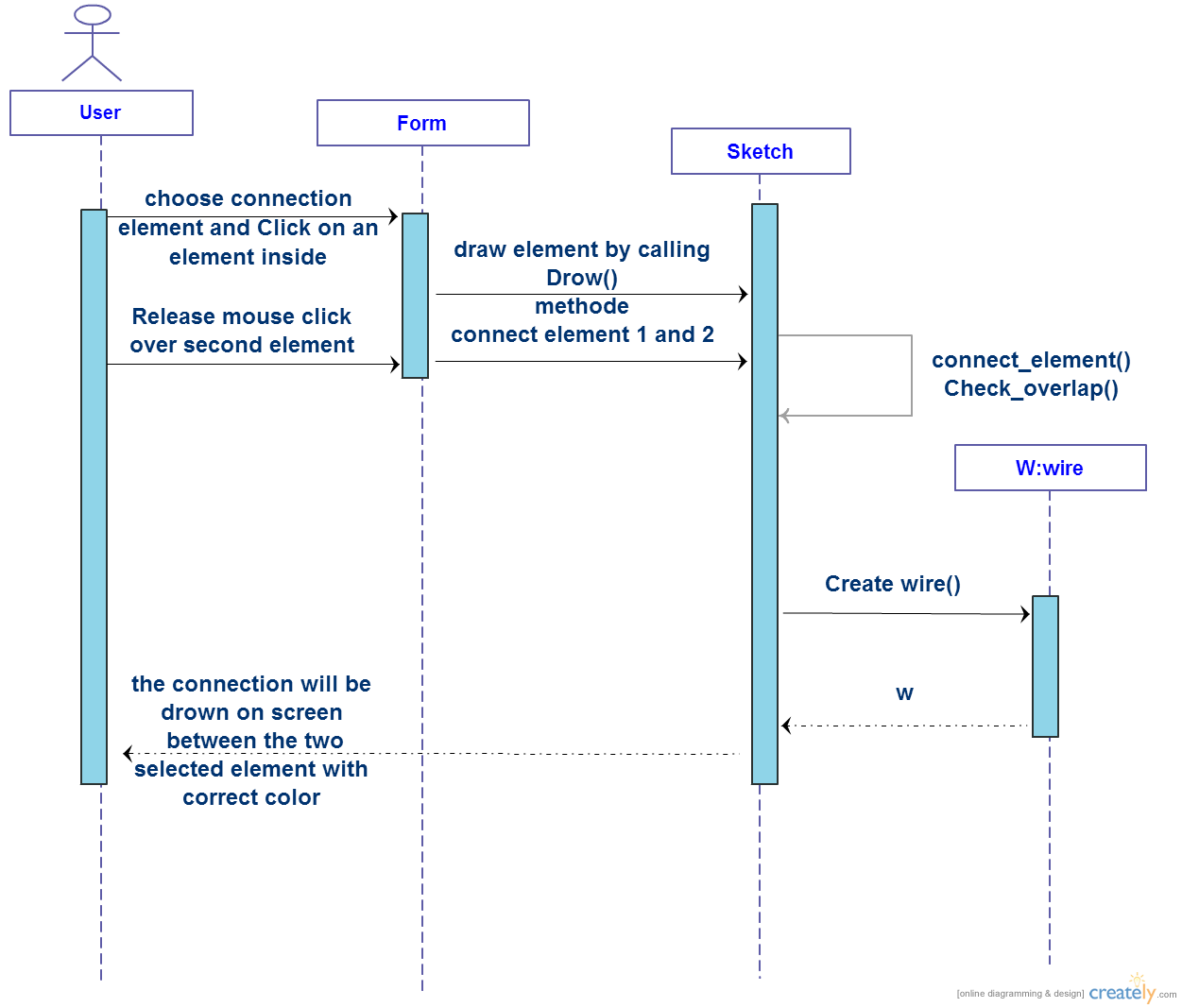
**Draw an element on the workspace area**



***Figure 5.1***

* In Figure 5.1 the sequence diagram that shows the interaction between the user and interface to successfully draw an element is shown. User can choose an element from the toolbox and select on a spot inside the workspace area after that the interface will create the required element and beside that it also checks for the overlapping. If everything goes well the element will be drown on the screen.

**Make a connection beetwen the two elements**



***Figure 5.2***

* In Figure 5.2 a successful connection between the two chosen elements is shown in the content of sequence diagram scenario.

User needs to click & hold, on an existing (drown) element inside the workspace area and then releases the mouse over the second element. Afterward the interface verifies the condition of the connection.

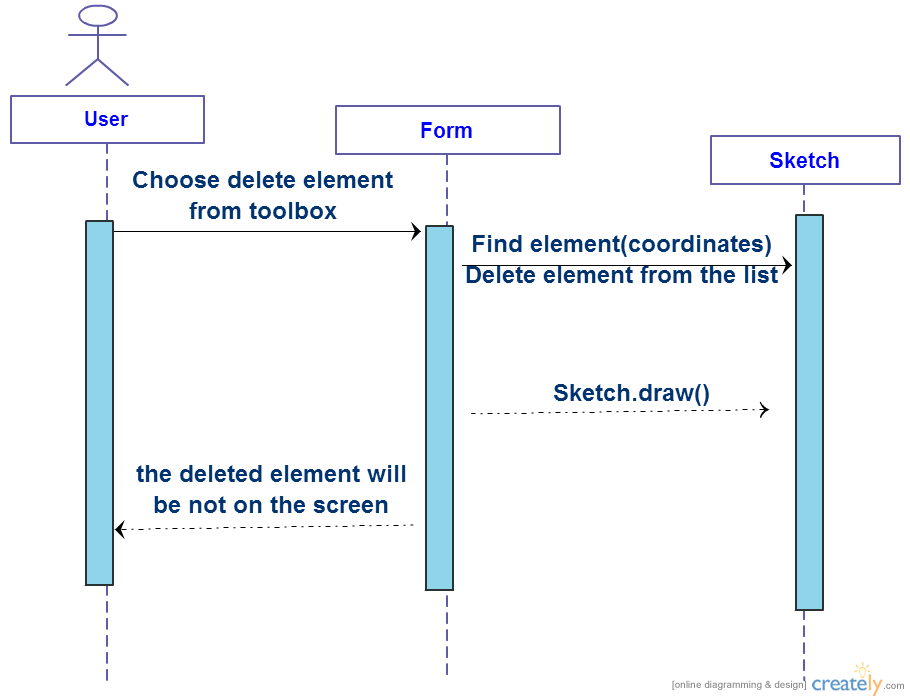
Every elements has one input but the output depends on the type.

The requirement for the connection is that the first elements need to have one free output and the second one free input.

-Interface will calculate the colour based on the value that the wire holds.

Besides all the overlapping between the ways of connection will be calculated and at the end the connection will be drown.

**Delete an existing element from the workspace area**

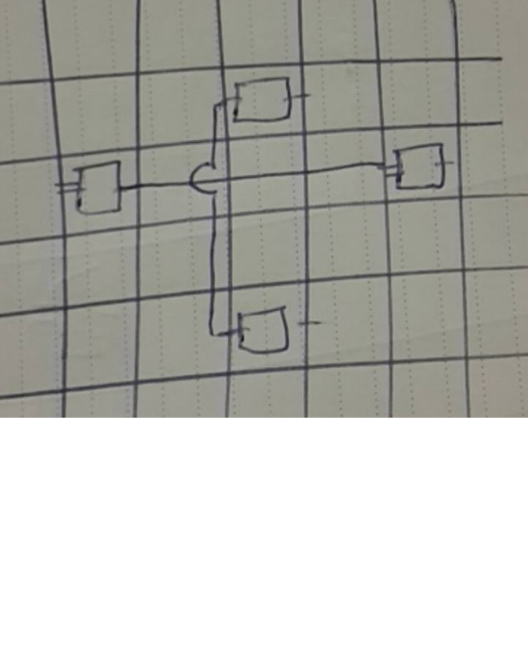


***Figure 5.3***

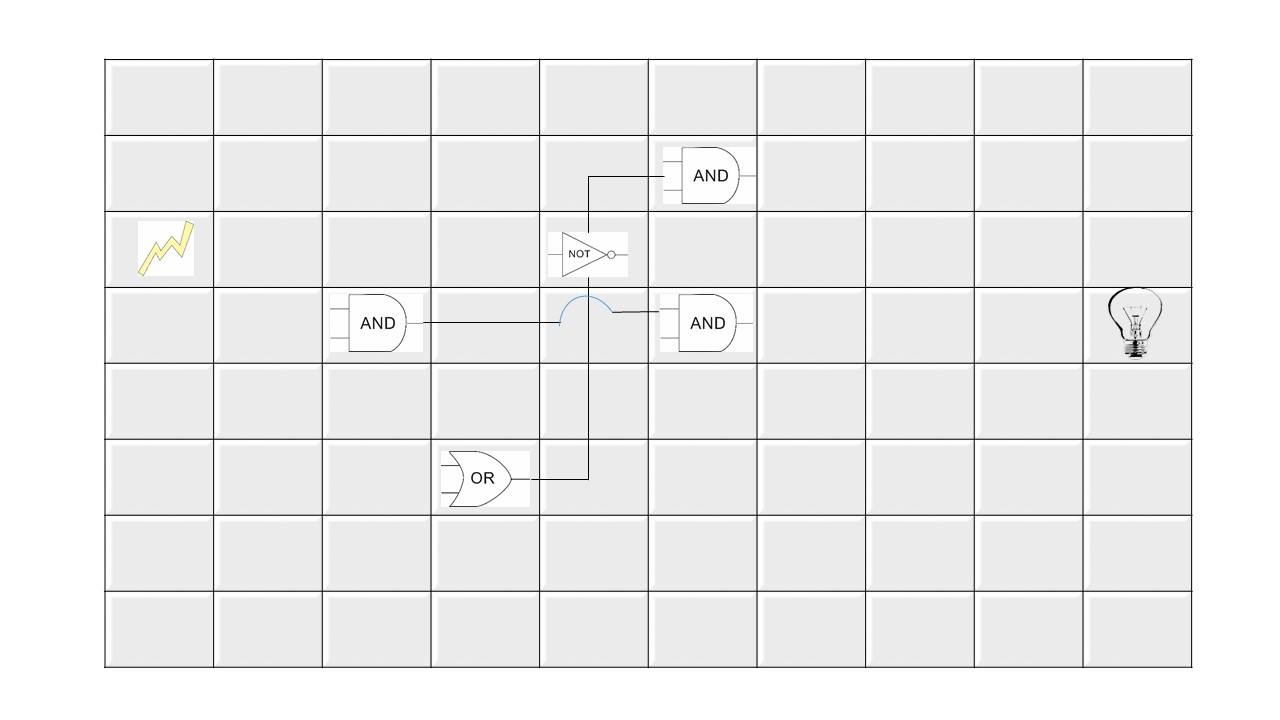
* Figure 5.3 shows the sequence diagram for deleting an existing element inside the current project workspace area. User can choose the delete action from the tool box and then click on an drown element inside the workspace area after that the interface will find the element base on the coordinates of the user's click spot and delete it from the list of the drown elemnts.At the end the update list will be drawn again.

1. **Assumptions/Risks**

* If there is a conncetion to be ovelapped in the way of the required connection then the program will automatically creates a bridge .See figure (6.1)
* In cases that user intends to draw an element but in that spot there is an existing element drawn already ,then an error would be shown in Error’s detail section.
* If there is a gate to be ovelapped in the way of the required connection then overlapping would be ignored and in this case the conncetion will simply cross the gate since its implementations are beyond the scope of this project. See figure (6.2)

******

***Figure 6.1***

******

***Figure 6.2***

**SaRistoKel Digital Circuit (SDC)**