

**Deriving a new Markup Language  
from the Extensible Markup Language (XML)  
to enhance Electrical Engineering Applications**

إشتقاق لغة ترميز جديدة من لغة الترميز الموسعة  
لتعزيز تطبيقات الهندسة الكهربائية

By

**Areej Yousef Abu Hadbah**

**Student's number: 401210083**

Supervisor

**Dr. Hebah. H. O. Nasereddin**

**Computer Information Systems Department**

**Faculty of Information Systems and Technology**

**Middle East University**

**This Partial Fulfillment is submitted as one of the Master's Degree  
Requirements in Computer Information Systems**

(2014)

## AUTHORIZATION FORM

إقرار تفويض

أنا أريج يوسف أبوهديبة أفوض جامعة الشرق الأوسط بتزويد نسخ من رسالتي للمكتبات  
المؤسسات، الهيئات و الأفراد عند طلبها.

الاسم : أريج يوسف إبراهيم أبوهديبة .

التاريخ : ١٣ / ١١ / ٢٠١٤


التوقيع : AR

## Authorization Statement

I'm authorizing the Middle East University to supply copies of my Thesis for Libraries, Establishments, Bodies, and Individuals upon their request.

**Name: Areej Yousef Abu Hadba.**

**Date:** 11/3/2019

**Signature:** 

## Examination Committee Decision

This is to certify that the thesis entitled "Deriving a new Markup Language from the Extensible Markup Language (XML) to enhance Electrical Engineering Applications" was successfully defended and approved on: **11 / 3 / 2014**

Examination Committee Member

Signature

1- Sadeq Al Hamouz



2- Hebah H.O. Nasereddin



3- Bayan Aref Abu Shawar



## **DEDICATION**

The Almighty Allah says “And remembers! Your Lord caused to be declared (publicly): "If ye are grateful, I will add more (favours) unto you; But if ye show ingratitude, truly my punishment is terrible indeed.” (Surah Ibrahim, Verse 7)

So all praise is for Allah, the exalted, for his favours that cannot be counted.

I dedicate this work to my husband, my parents, my daughter, my family, my sisters, my relatives, my friends, and to all those who helped, supported and taught me.

## **ACKNOWLEDGMENT**

My endless gratitude to my husband who is struggling to make my family life easier, my allegiance and sincerity are for you.

My great thankfulness to my supervisor Dr. Hebah. H.O. Nasereddin for her supports, encouragements, giving help throughout this period, and for scrutinizing of my thesis draft.

I dedicate this work to my parents, my daughters Yara and Lara, my sisters, brothers, relatives, friends, and all who have supported me.

I would like to appreciate all the Information Technology Faculty Members of the Middle East University specially, Dr. Ahmad kyed and Dr. Maamoun Ahmed for their assistance which I had been received since I started this thesis.

## Table of Contents

Deriving a new Markup Language From (XML) to enhance Electrical Engineering Applications.....	I
Authorization Statement.....	III
COMMITTEE DECISION.....	IV
DEDICATION.....	V
ACKNOWLEDGMENTS.....	VI
<b>Table of Contents.....</b>	<b>VII</b>
List of Tables.....	IX
List of Figures.....	X
List of Abbreviations.....	XIII
الملخص.....	XVII
<b>ABSTRACT.....</b>	<b>XIX</b>
<b>Chapter One: Introduction.....</b>	<b>1</b>
1.1 Overview.....	1
1.2 Problem Statement.....	4
1.3 Contribution.....	5
1.4 Motivation.....	5
1.5 Thesis Objectives.....	6
1.6 Thesis Structure.....	6

<b>Chapter Two: Literature Survey and Related Works.....</b>	<b>7</b>
2.1 Overview.....	7
2.2 Related Works.....	7
2.3 XML Technologies and Application.....	18
2.4 Software Programs.....	20
2.5 Web Site Application.....	21
<b>Chapter three: Designing the Proposed Model.....</b>	<b>22</b>
Overview.....	22
3.1 Proposed Model.....	23
3.1 Creating a New Markup Language for Electrical Engineering.....	24
3.1.1 Classifications Tables of Electrical Symbols.....	24
3.1.2 Syntax and the XML Rules.....	28
3.2.3 Writing the XML Codes.....	31
3.2.4 XML Parsers.....	35
3.2.5 Connecting the Elements together.....	36
3.2.6 XML Schema definition (XSD).....	39
3.3 Designing the web application which supports the New Language.....	42
3.3.1 Write the Descriptive Code in Text Box.....	43
3.3.2 Writing The Function and The Procedure.....	50
<b>Chapter Four Experimental Results.....</b>	<b>70</b>
4.1 Compare and Results.....	70



4.2	Draw The Same Circuit on Software Program and Proposed Model.....	80
4.3	Limitations.....	85
<b>Chapter five: Conclusion and Future Work.....</b>		<b>86</b>
5.1	Overview.....	86
5.2	Conclusion.....	86
5.3	Future Works.....	87
	References.....	88
	APPENDICES.....	95

### List of Tables

Table (3.1):	Wires Symbols.....	24
Table (3.2):	Switches Symbols and Relays Symbols.....	25
Table (3.3):	Ground Symbols.....	26
Table (3.4):	Resistors Symbols.....	26
Table (3.5):	Logic Gates Symbols.....	27
Table (4.1):	Comparing the Based on the size Image.....	72
Table (4.2):	Comparing the Based on the size image on Disk.....	74
Table (4.3):	Comparing the Based on the Requirements.....	74
Table (4.4):	Comparing Based on the Plate Form Independent.....	75
Table (4.5):	Comparing Based on the Edit on Circuit.....	76
Table (4.6):	Comparing Based on the Extract Information From Drawing.....	76
Table (4.7):	Comparing Based on the Media Type.....	77

Table (4.8): Comparing Based on the Accuracy With Zoom.....	77
Table (4.9): Comparing Based on the Sharing and Exchange Information.....	78
Table (4.10): Comparison Table.....	79

## List of Figures

Figure (2.1): (MathML) Code.....	9
Figure (2.2): Web Application Base (MathML) in Arabic.....	9
Figure (2.3): Web Application Base (MathML) in English.....	10
Figure (2.4): CML Code represents Molecules and Chemicals Information.....	12
Figure (2.5): CML Application Based on XML Language.....	12
Figure (2.6): (WML) Code Based on XML.....	14
Figure (2.7): The Difference Between Bitmap and Vector Images.....	15
Figure (2.8): (BML) Code.....	16
Figure (2.9): X3D Code Based on XML.....	17
Figure (3.1): Proposed Model.....	23
Figure (3.2): Images are Used in the Proposed Model.....	28
Figure (3.3): (BAT) Element has an Open and close tag.....	29
Figure (3.4): Write Value in XML and HTML.....	28
Figure (3.5): Error Code for Case Sensitive.....	30
Figure (3.6): The Ordered tags in XML and HTML.....	30
Figure (3.7): Root and Child Element.....	31
Figure (3.8): Declaration the XML Document.....	32
Figure (3.9): Create the Root Element (VOC).....	32

Figure (3.10): Create Child Element <Elements>..... 33

Figure (3.11): Tree Structure for the XML Document..... 34

Figure (3.12): The XML Tree for the Proposed Model..... 35

Figure (3.13): The XML Code represents an Element has One Connection Point..... 36

Figure (3.14): The XML Code represents an Element has Two Connection Points..... 37

Figure (3.15): The XML Code represents an Element has Three Connection Points..... 38

Figure (3.16): The XML Code represents an Element has Four Connection Points..... 38

Figure (3.17): The XML Schema Definition for Electrical Markup Language..... 40

Figure (3.18): (XSD) Elements..... 40

Figure (3.19): Data Grid for XML Document..... 41

Figure (3.20): The Description of the Structure Code..... 44

Figure (3.21): Table represents the Numbers Position on the Screen..... 45

Figure (3.22): Equation for Finding (x, y) Point Automatically..... 45

Figure (3.23): The (x, y) Point is Located in Position ..... 46

Figure: (3.24): The Figure Explains the Element Connection sides..... 47

Figure: (3.. 25) Descriptive Code..... 47

Figure (3.26): A Circuit Contains only Horizontal Elements..... 48

Figure (3.27): An Equation is used to Locate the Point After Rotating the Element..... 49

Figure (3.28): The point Location After rotating the Element..... 49

Figure (3.29): The Check Node Function..... 50

Figure (3.30): The Check Node Error..... 51

Figure (3.31): Loading Procedure of the XML File and Checking the Root tag..... 52

Figure (3.32): An Error has Happened Because of forgotten the Root tag by the User..... 52

Figure (3.33): ID Checking Function..... 53

Figure (3.34): A Page was Loading by Session..... 54

Figure (3.35): Send data from (SVG) to draw the Page.....	54
Figure (3.36): Draw Circuit Function.....	55
Figure (3.37): Calls Shape Function.....	56
Figure (3.38): Function to Draw Lines.....	57
Figure (3.39): Get Side Function.....	57
Figure (3.40): Help Page.....	59
Figure (3.41): Search Page.....	60
Figure (3.42): Result for Search about (bat) word.....	61
Figure (3.43): Code to order the result search in table.....	62
Figure (3.44): Procedure to get Picture in cell (5).....	62
Figure (3.45): function to find number the left and right edges for every element.....	63
Figure (3.46): rotates element.....	63
Figure (3.47): Explain rotates element.....	64
Figure (3.48): Send (SVG) data in text box (cc).....	64
Figure (3.49): Script to call code mirror.....	65
Figure (3.50): Main screen has code written by the user.....	66
Figure (3.51): Electrical circuit in draw page.....	67
Figure (3.52): Cross line problem.....	68
Figure (3.53): Added element to solve problem.....	69
Figure (3.54): (PLANK) image.....	69
Figure (4.1): Basic circuit drawn with all programs.....	71
Figure (4.2): Compares between software programs with proposed model base numbers.....	73
Figure (4.3): Express PCB program.....	80

Figure (4.4): Electrical circuit was drawn by Express PCB.....	80
Figure (4.5): Electrical circuit drawn by Circuit Maker 2000.....	81
Figure (4.6): Electrical circuit drawn by Circuit Maker 2000.....	81
Figure (4.7): program is SEE Electrical LT.....	81
Figure (4.8): Electrical circuit drawn by (SEE Electrical LT).....	82
Figure (4.9): Electrical circuit design with our project.....	83
Figure (4.10): XML code written by user to draw the circuit.....	84
Figure (4.11): electrical drawing of our project.....	85

### **List of Abbreviations:**

(SVG):	Scalable Vector Graphics
(XML):	Extensible Markup Language
(VOC):	Vocabulary
(Automation ML):	Automation Markup Language
(X3D):	Extensible 3D Language
(XBRL):	Extensible Business Reporting Language
(BIPS):	Bank Internet Payment System
(EBXML):	Electronic Business XML Initiative
(CXML):	Commerce XML
(PCB)	Printed Circuit Board
(OGC):	Open Geospatial Consortium
(THML):	Hypertext Markup Language

(CSS):	Cascading Style Sheets
(MathML):	Mathematical Markup Language
(CML):	Markup Language
(WML):	Wireless Markup Language
(W3C):	World Wide Web Consortium
(SGML):	Standard Generalized Markup Language
(DSML):	Directory Services Markup Language
(CDF):	Channel Definition Format
(BXXP):	Blocks Extensible Exchange Protocol
(XUL):	Extensible User Interface Language
(DTD):	Data Type Definition

## الخلاصة

ان لغات الترميز النصي المشتقة من لغة الترميز القابلة للامتداد تم تطبيقها في العديد من المجالات منها الرياضيات، الكيمياء، الطب، المكتبات الالكترونية والنشر الالكتروني . في هذه الرسالة طرقتنا مجالاً جديداً: إشتقاق لغة ترميز نصيه دُعمت بتطبيق انترنت لتمكين المهندس الكهربائي من رسم الدوائر الكهربائيه .

تمتاز الدوائر الكهربائيه المرسومه بهذه اللغه من خلال التطبيق الجديد بالمقارنه مع برامج الرسم التقليديه بسهولة رسمها والتعديل عليها وتبادلها عبر الانترنت كونها تخزن كملفات نصيه وليس على هيئة صوره.

إن الميزة الفريدة في الدارات تبادلها ونقلها وتخزينها يتم عبر الملف النصي وليس على شكل صورة , مما يوفر فرقاً كبيراً في حجم الملفات المنقولة مقارنة مع البرامج المتخصصة بالرسم ، بالإضافة لذلك إمكانية التطبيق على كل منصات وأنظمة التشغيل ، وتتميز الدارات المرسومة وفق هذا التطبيق بسهولة التعديل عليها بالمقارنه مع الدارات المرسومة بالبرامج الهندسية ,ويمكن لمحركات البحث إستخراج البيانات من تلك الدارات عبر الويب لأنها تنقل على هيئة ملف نصي وليس على هيئة صور ، وقد تم الرسم في هذا التطبيق ضمن بيئة الرسومات المتوجهة القابلة للإمتداد والتوسع ، وهي تقنية تستخدم لتوصيف الرسوم بدلا من عرضها على شكل ( بيكسيل ) , وتتميز هذه التقنية بالقدرة على تكبير وتحجيم الصورة لأي مدى دون فقدان أي من تفاصيل الصورة وبالتالي الحصول على نفس دقة الصورة كما تبدو على الشاشة حتى بعد طباعتها مرات عدة.

## **Abstract**

This study aims to derive a new Markup Language from Extensible Markup Language (XML), which was applied in many fields like mathematics , chemistry , medicine, electronic libraries and electronic publication . In this model various filed is opened by deriving a new text markup language and creating a new web application in ordered to enable electrical engineers to draw electrical circuits. Many advantages are done when this language and application are used to create drawings instead of using the ordinary drawing programs, such as saving a great deal of stored size of data which are transformed on the world wide web as text form instead of images.

The main unique character of these circuits is that they are exchanged, transformed and saved through a text file not in the form of an image, saving a great deal of the size of the files which were transformed and the amount of storage comparing with the professional drawing programs.

In addition to that, The ability to apply this model on all platforms and operating system. The drawn circuits by using this application are easily modified comparative with the drawn circuits by using professional drawing programs. Browsers (searching engines) can extract data from those circuits through the web because they are transformed as text format not as image format.



# Chapter One

## Introduction

As a result of massive developments in technologies, the requirements for sharing and exchanging data over the web are rapidly increasing. These incredible global desires for communicating information are forcing the programmers to develop ways to match this accelerated passion of the customers; one of these ways is contriving new languages.

When you transfer and share data over the web , speed , flexibility, and loading space are important issues. Many researches direct to XML language and customize it in their projects. XML (Extensible Markup Language) is designed to facilitate the ability to transfer data and to share information with others over the web. In addition , it provides the ability to use several applications to get important advantages.

(Jakub Klímek, 2012) XML is a standard for communication in various information systems, where it is used to derive other languages to achieve certain goals in different fields, such as math and chemistry. After success in this language and applications, the researcher turns to research in XML, to know how many benefits that could be gained from it in our life. (H. M. Deitel, 2001) “XML is a flexible way to create common information formats and share both the format and the data on the World Wide Web, XML formal

Recommendation from the World Wide Web Consortium (W3C) and derived from SGML (ISO 8879).” XML is like (HTML) Hypertext Markup Language; both of them contain markup symbols used to describe the contents of a page. But there are fundamental differences between them; (Berlin, 2014) HTML designed to describe and display the content of a web page's text as how it is displayed, and to focus on how data look.

The XML concern of the content in terms of what data are being described, transported, stored and carried. XML is an important tool for data transmission between applications and it allows users to set standards defining the information that you need. XML provides a basic standard that can be used to share information between different applications in different organizations. XML is an important language for people who are interested in obtaining language that provides facilitates in exchanging the at high speeds with accuracy and safety at the same time. (Das, 2005) “SGML and HTML are not good for this purpose, HTML and SGML provides arbitrary structure, but is too difficult to be implemented just for a web Browser”.

(Peter, 2003) XML is a flexible language. By XML you can mark up and create your own tags and attributes, so you can use markup that actually describes the content of the element rather than just using tags that tell you how to present the data on a web page. As you can use tags that describe the content of an element, XML has become a general format for marking up all kinds of data, not just data that will be presented on the web.

(walsh, 2000) "XML is Meta language which means it's a language that allows you to create your own markup languages. Unlike HTML, XML is meant for storing data, not dispelling it".

(IBM, 2009) XML provides a way of structuring your data in documents, the reason it's taken off so quickly so it's perfect for the internet. Since XML documents are texts, you can send them using the existing internet technology that was built in HTML. (Tom Myer, 2005) . Users can send data over the internet, to people or dedicated software. XML is designed to help store, structure and transfer data; (Holzner, 2003) "because it's written using plain text, it can be sent on the internet and handled by software on many different platforms". Briefly, XML is designed to allow people circulate their data.

First, data is extracted from an (XML) documents, then working with this data when opening a project on the internet, the explorer displays a button that when clicking the button, the Java script or (vb.net) reads each element in our sample XML document. In this way we have been able to extract data from an XML document, also we can send the document over the internet, and you can extract the data you need from the document by searching for elements with specific names. (Holzner, 2003) In XML, data "is stored as text that you yourself can configure if something goes wrong, you can examine or modify the document directly because it's all just text".

So, languages derived from XML help to solve problems in particular areas such as (MathML). MathML is used to facilitate the reuse of mathematical and scientific content on the Web. There are many languages derived from XML such as XSLT (Extensible

Stylesheet Language Transformations), XPath, XQuery, XLink, XSchema, XSL-FO, SVG, APML (Attention Profiling Mark-up Language), CXML, (EBXML), a collection of Electronic Business specification, (Music XML) and (OPML) an XML format for outlines.

The aim of this thesis is to derive a new markup language based XML language. This language is derived to support electrical engineers and helps them to draw their electrical circuit over the web. The implemented web application shows the importance of the new language and what facilities provided to the electrical engineering users.

## 1.2. Problem Statement

JavaScript XML provides a simple format that is flexible enough to accommodate diverse needs. XML can solve many problems in any field and provide facilities when using it. In this research we will use this advantage to solve some problems facing electrical engineering and discuss the problems facing engineers when browsing the web especially in their research and their electrical circuit design. The engineers can't draw their scheme on the web directly; they should install the software programs to draw. After drawing the electrical circuit by software programs, the outputs are represented as images.

These images have a large size that also can't be shared with others easily over the web, some problems emerge when loading or sending schema, because it is sent as an image; the search engines can't extract information from these images. If engineers want to modify any element in the image, they should redraw it again. and solve the platform dependency problem.

### **1.3. Contribution**

**This thesis contributes in the following findings:**

- 1- Derives a new markup language base XML; this new language will be designed for engineering electric circuit.
- 2- Supports this language with a new application to use a new language on the web and drawing circuit.
- 3- Explains how this new language helps to support and provide facility to engineer.

### **1.4. Motivation**

When we use XML in any subject, we can get many features and solve many problems and provide many facilities to users. XML browsers allow users to read XML documents freely. Briefly, this language is interesting in any field, in which it is used. When started thinking about the domain that wanted to derive a markup language for it, this language for electrical engineering was chosen.

The main motive to choose this field is that my husband is an engineer, and I talked to him about some of the problems he faced while browsing the Web, and what problems facing engineers when using a software program for drawing. So, this thesis is directed to using (XML) to solve some of the problems in electrical engineering.

## **1.5. Thesis Objectives**

The following are the objectives of this thesis:

1. The main objective of this research is to derive a new markup language base XML; this new language will be designed for electrical engineering to provide more facility and flexibility when drawing circuits.
2. It helps engineers to design their schema with more flexibility and more interactive way in less space and load on storage, because the schema is sent as a text not as an image.
3. It provides this language with a new application of engineering to use a new language on the web, because we can use this language for multiple applications.

## **1.6. Thesis Structure:**

The thesis includes five chapters; the current chapter is the introduction. Next chapter is about literature and a survey for thesis, showing the related work regarding XML languages. The design of the proposed model of a new markup language base XML and supporting this language with web site application are presented in chapter three. Then we present the comparison and results of experiments in chapter four. Finally conclusion and the future work for the thesis are discussed in chapter five.

## Chapter Two

### Literature Survey and Related Works

#### 2.1 Overview.

XML is a Meta language, because it can create sub languages from it. This section we will discuss notable markup languages, that have been created using XML. There are many languages designed based on XML language, all these languages add advantages in many fields and solve many problems. (Martin Erwig, 1999) XML sub languages are called XML applications, allowing various groups of people to communicate and exchange data.

#### 2.2 Related Works.

(W3C) World Wide Web Consortium, recommended (MathML) on (2010). “Mathematical markup language (MathML) was designed to let people embed mathematical and scientific equations in web pages”. With (MathML) you can display all kinds of equations.

(H.M.Deitel, 2001) “(MathML) is an XML application for describing mathematical notation and capturing both its structure and content. The goal of (MathML) is to enable mathematics to be served, received, and processed on the Web, just as HTML has enabled this functionality for text”.

(Stephen Buswell, 7 July 1999) Mathematical expressions have typically been displayed using images. (MathML) is describing mathematical excretions using XML syntax. The W3C provides a browser called (Amaya) to edit, parse and render (MathML).

There are some Problems (MathML) tries to solve. (Iion, 1998) next example for equation to show in the web is represented as image  $2^{2^x} = 10$ . This equation has a particular size if a researcher wants to copy this equation as an image that facing many problems.

The first problem is if the user copies equation as image in another paper, the font's lines forms a problem because of that difference. The second problem is if the equation image was created by the browser or user with another color, the anti-aliasing in the image produces white "halos."

$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ . Another problem about size image after copy and paste it in

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

another place, maybe appear like this:

Another problem when showing mathematics equations as images is that it requires more bandwidth. When using a markup language (MathML), a large amount of the process is moved to the client machine. (MathML) describes an equation smaller and more compressible than an image of the equation.



(David Carlisle (NAG), 2000) The various languages and formats that make up modern web pages (MathML of TML, XHTML solution SS) work on finding solutions a web browser problems in web browsers.

MathML allows the user to write his equation using XML code and solve all Previous problems, such as the example in the next figure (2.1):

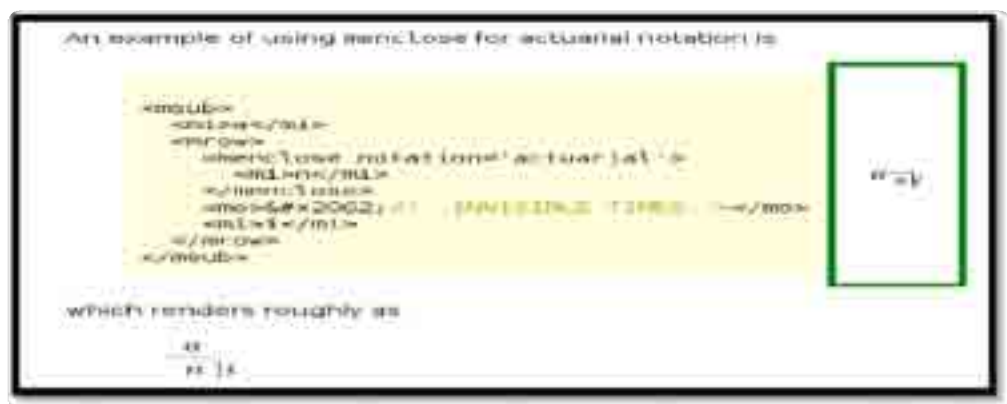


Figure (2.1): (MathML) Code.

Many researcher designed web application base (MathML) such as Arabic mathematics in mathzilla. (Fredw, 2011).( Arabic mathematics in Mathzilla).



Figure (2.2): Web application base (MathML) in Arabic

(fredw, 2011) Math Parser adds a component to parse mathematical expressions into (MathML). It requires Math Parser to be installed and provides the features that people are interested in (MathML).

A tiny (MathML) editor, It's basically an interface for Math Parser, with an input and output field. Figure (2.3) this allows copying the (MathML) outputs of previous features.



**Figure (2.3): Web application base (MathML) in English.**

Chemical markup language (CML) with (Wibe, 2013) CML can view three dimensional representations of molecules. With CML chemist can publish a molecule and exchange that model with others. With CML, users can display the structure of complex molecules.

(Rzepa, 2001) “(CML) is presented as an XML Schema compliant form, modularized into non chemical and chemical components. While (CML) Core retains most of the chemical functionality of the original (CML 1.0) and extends it by adding handlers for chemical substances, extended bonding models and names”.

(Murray-Rust P, 2003) CML is a language derived from XML language for representing molecular and chemical information. CML takes “advantage of XML’s portability to enable authors document to use and reuse molecular information without corrupting important data in process”.

The figure (2.4) represents CML code write for representing chemical information

```

--<xsd:schema xmlns:is="http://www.w3.org/1999/xhtml" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.xml-cml.org/schema" targetNamespace="http://www.xml-cml.org/schema">
  <xsd:simpleType id="st.actionOrderType" base="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        <h:div class="summary">
          Describes whether child elements are sequential or parallel.
        </h:div>
        <h:div class="description">
          There is no default.
        </h:div>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="sequential" />
      <xsd:enumeration value="parallel" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType id="st.alternativeTypeType" name="alternativeTypeType">
    <xsd:annotation>
      <xsd:documentation>
        <h:div class="summary">
          The type of an alternative.
        </h:div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:simpleType>

```

Figure (2.4): CML Code represents Molecules and Chemicals Information.

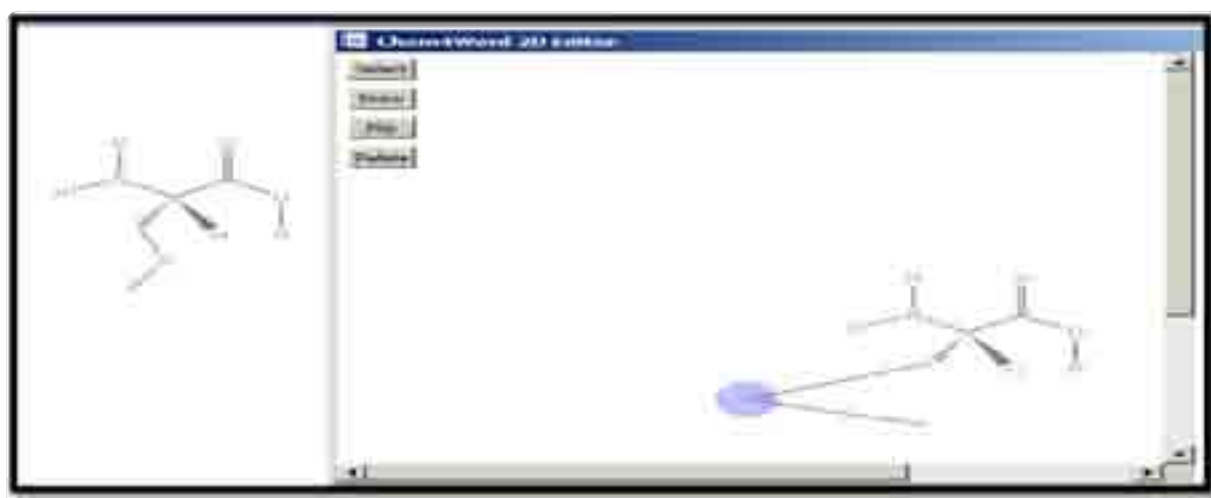


Figure (2.5): CML application base XML language.

(Peter Murray-Rust, 2000) “The operating system for managing complex chemical content entirely in interpreting XML based markup languages. And the development of

mechanisms allowing the display of CML marked up molecules within a standard web browser (Internet Explorer 5)".

Wireless markup language (WML). (Kumar. M, 2013) WML is a markup language base XML used to specify content and user interface for WAP devices like Mobile Phones. This language allows web pages to be displayed on wireless devices.

(H.M.Deitel, 2001)"WML works with the wireless Application protocol WAP to deliver content. WML is similar to HTML, but it does not require input devices such as a keyboard or mouse for navigation. A WML document is called a deck and contains static parts called cards. Each card consists of one page of information, providing the WML browser with a small, self-contained document for browsing". This packaging of multiple pages of information is necessary because the device to which WML is delivered has limited connections to the Internet.WML includes a telephone. A voice mail service can have a WML user interface that gives the users choices about their mailbox.

Any WML document will be represented properly on a WAP compliant device, as micro browsers will automatically tailor the WML presentation to the specific device. WML documents are XML documents that validate against the WML (DTD).

Figure (2.6) shows (WML) page that could be saved as example of WML:

```
<? xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
  <card id="main" title="First Card">
    <p mode="wrap">this is a sample WML page. </p>
  </card>
</wml>
```

**Figure (2.6): (WML) code base XML.**

Scalable vector graphics (SVG) (Mohan, 2010) “The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999”.

The scalable vector graphics (SVG) is a markup language base XML language used to create vector graphics data over the web. SVG provides an advantage rather than current methods such as JPG, GIF and (png).

(Mohan, 2010) The current formats for image used bitmaps, which describe each pixel in the image and may take more time to download .Bitmap resolution, is fixed, so bitmap images cannot be scaled (zoomed, panned, etc.) Without a loss in image quality, and printed bitmap images often contain low resolution, jagged lines. (Etherington, G, 2013) Conversely, vector graphics describe graphical information in terms of lines, curves. Not only images rendered in vector graphics require less bandwidth, but these images also can be easily scaled and printed without producing jagged lines.

Because SVG is an application of XML, SVG document can be scripted, searched and dynamically created. SVG images defined in text files. So these images can be searched, indexed, scripted, and compressed. As XML files, SVG images can be created and

edited with any editor. Figure (2.7) shows the difference between regular image and (SVG) image.



**Figure: (2.7) difference between bitmap and vector images. (www.Wikipedia.com)**

(Zavlavsky, I. 2013) The main difference between SVG vector and bitmap images is that the bitmap image is composed of a fixed set of dots, while the vector image is composed of a fixed set of shapes.

(Hundt, 2008) discussed the Automation ML.(Automation Markup Language) is a neutral data format based on XML for the storage and exchange of plant engineering information, which is provided as an open standard. The goal of“(AutomationML) is to interconnect the heterogeneous tool landscape of modern engineering tools in their different disciplines, e.g. mechanical plant engineering, HMI (Human Machine Interface development), PLC (Programmable Logic Controller), and robot control”

(Death, 2008) Automation ML (Automation Markup Language) serves for the data exchange between manufacturing engineering tools and therefore supports the

interoperability between them. It covers information about the plant structure (topology and geometry) and the plant behavior (logic and kinematics).

(aktualisiert, 2013 ) “Broadcast Markup Language, or BML, is an XML-based standard developed by Japanese ARIB association as a data broadcasting specification for digital television broadcasting”, figure (2.8) describes an example for BML language

```
<?xml version="1.0" encoding="EUC-JP"?>
<!DOCTYPE bml PUBLIC "-//ARIB STD-B24:1999//DTD BML
Document//JA" "bml_1_0.dtd">
<? bml bml-version="1.0"?>
```

**Figure (2.8): (BML) code.**

Extensible 3D language (X3D) in 1997, the Web 3D Consortium recommended the Virtual Reality Modeling Language (VRML97) for “use on the Internet as a file format for describing interactive 3D objects and worlds”.

(Huang, 2003) Virtual world’s three diminution objects grouped together in some common way. X3D is recommended of the World Wide Web consortium and the Web3D Consortium. (Liu, S, 2013) X3D is the next generation of VRML, and the current version is backwards compatible with VRML97. X3D’s XML base figure (2.9) gives example for X3D.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "-//Web3D//DTD X3D-1.2//EN"
"http://www.web3d.org/specifications/x3d-1.2.dtd">
<X3D profile="Interchange" version="1.2"
xmlns:xsd="http://www.w3.org/2001/XMLSchema-
instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/spe
cifications/x3d-1.2.xsd">
<Scene>
<Shape>
<IndexedFaceSet coordindex="0 1 2">
<Coordinate point="0 0 0 1 0 0 0.5 1 0"/>
</IndexedFaceSet>
</Shape>
</Scene>
</X3D>
```

Figure (2.9): X3D code base XML.

(Debreceeny, 2005) "The XBRL Specification is developed and published by XBRL International".

(Hamscher, 20 April 2004) Extensible Business Reporting Language (XBRL). Development of the (XBRL) began in 1998, initially under the direction of the American Institute of Certified Public Accountants (AICPA). (Florin, D, 2013) XBRL captures existing financial and accounting information standard in XML, providing significant advantages over current methods of business information representation and transfer. (Hamscher, 20 April 2004) "XBRL's XML origin permits financial information to be reused and a variety of situations (e.g publishing reports, extracting data for applications, and submitting regulatory forms), thus, increasing efficiency and reducing cost and redundancy".

(Sprott, 2000) "Electronic business XML initiated (EBXML) In September 1999, the United Nation's Center for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the advancement of structured information standards (OASIS) to standardize the global exchange of business information". The result of this project is Electronic Business XML (EBXML).

Commerce XML (CXML) is an XML based framework for describing catalog data and performing business-to-business electronic transactions that use the data. It was developed by a group of 40 e-commerce companies, led by (Ariba). (CXML) enables businesses and suppliers to conduct transactions over the internet more efficiently. (CXML) provides several DTDs that describe data for catalogs, interactive catalogs and purchase orders. The language also specifies ways in which (CXML) documents may be requested and posted over the web.

(H.M.Deitel p. , 2003) Open eBook Publication structure. Several companies dedicated to the electronic text publication founded the open eBook Forum with the goal of developing a standard for describing publishable electronic content. The result of this collaboration is the open eBook publication structure, an XML based language that specifies a file format and describes content elements. (Beezer, J, 2013) The language is designed to be platform independent, but maintains flexibility and permits document authors to embed platform-specific content as long as “ a platform-independent alternative is provided. The Open eBook Forum hopes that wide adoption of the Open eBook Publication structure will spur growth in the electronic-publications market”.

### **2.3 XML Technologies and Applications:**

(H.M.Deitel p. , 2003) “XML Query Language (XML Query) uses the power of XSL patterns to search XML documents for specific data. As SQL (Structured Query Language) searches for data stored in relational database”. XML Query searches for data stored in an XML document. XML Query syntax resembles the path specified in a

UNIX environment. XML Query was submitted to W3C as a proposal in 1998 and development of the language is ongoing.

(Harvey, R, 2013) “Directory Services Markup Language (DSML) provides a method for managing relational resources and metadata”. Aside from their usual usage for storing records of organizational assets, (Pal, S, 2013) “directory services can be used with XML to dynamically match data across networks. The Directory Services Markup language (DSML) is the bridge between directory services information to be described in an XML document”. With DSML, directories gain the ability to handle distributed Web based applications, such as those used in e-business, network and supply chain management.

Channel Definition Format (CDF) is a Microsoft XML application implements a technology called push technology which automatically sends the contents to users. Using CDF, Web authors can define channels that automatically deliver content to subscribed users. For example, a user can subscribe to a financial-information Web site’s channel to receive information about certain stocks at a regular interval. Additionally, several related web sites can use channels to refer users between sites.

(H.M.Deitel p. , 2003) “Blocks Extensible Exchange Protocol (BXXP) is an alternative to HTTP for transferring data over the internet. BXXP was developed by Marshall Rose as a general application protocol on which users can develop more specific protocols for instant-messaging, chat or file-transfer applications. BXXP sends “blocks” of XML data over TCP and has the capability to send multiple simultaneous blocks of data. Currently, BXXP is still in development”.

(H.M.Deitel p. , 2003) Extensible User Interface Language (XUL) XML has proven a valuable catalyst in the field of cross-platform (XP) application development Project such as Mozilla work to provide cross-platform development platforms and services for use into a variety of purposes. One challenge facing cross-platforms, application developers is that each operating system has a unique graphical user interface (GUI).

## **2.4 Software programs.**

(Therapy, 1997) describes the Schematic Design Software .The (ExpressSCH) schematic design, program design are to be used as PCB (Printed circuit board) layout software, programs having user interface. In this program, the user should download the program in its computer that requires 10 MG space then start to draw by drag dropping the element on work space. The drawing output from this program exports as an image (bitmap pixels) . This program is easy to use, but has some problems represented by the large space for our image. Such as simple electrical circuit drawing requires 100 KB (102,400 bytes) to save. Also, with this program, there are difficulties to modify the image and drawing effectively to scale, zoom). Additionally, these programs are not independent platform.

## **2.5 Web site application.**

There are some web sites used for drawing electrical circuit online. This includes Circuit lab and Do circuits and (xcircuit). This site add advanced for users, the users don't need install the program on their computers.

The difference between this site and proposed model is the form of circuit after drawing on this site the circuit represents as image, but with proposed model the circuit represents as XML file (text file).

## **Chapter Three**

## DESIGNING THE PROPOSED MODEL

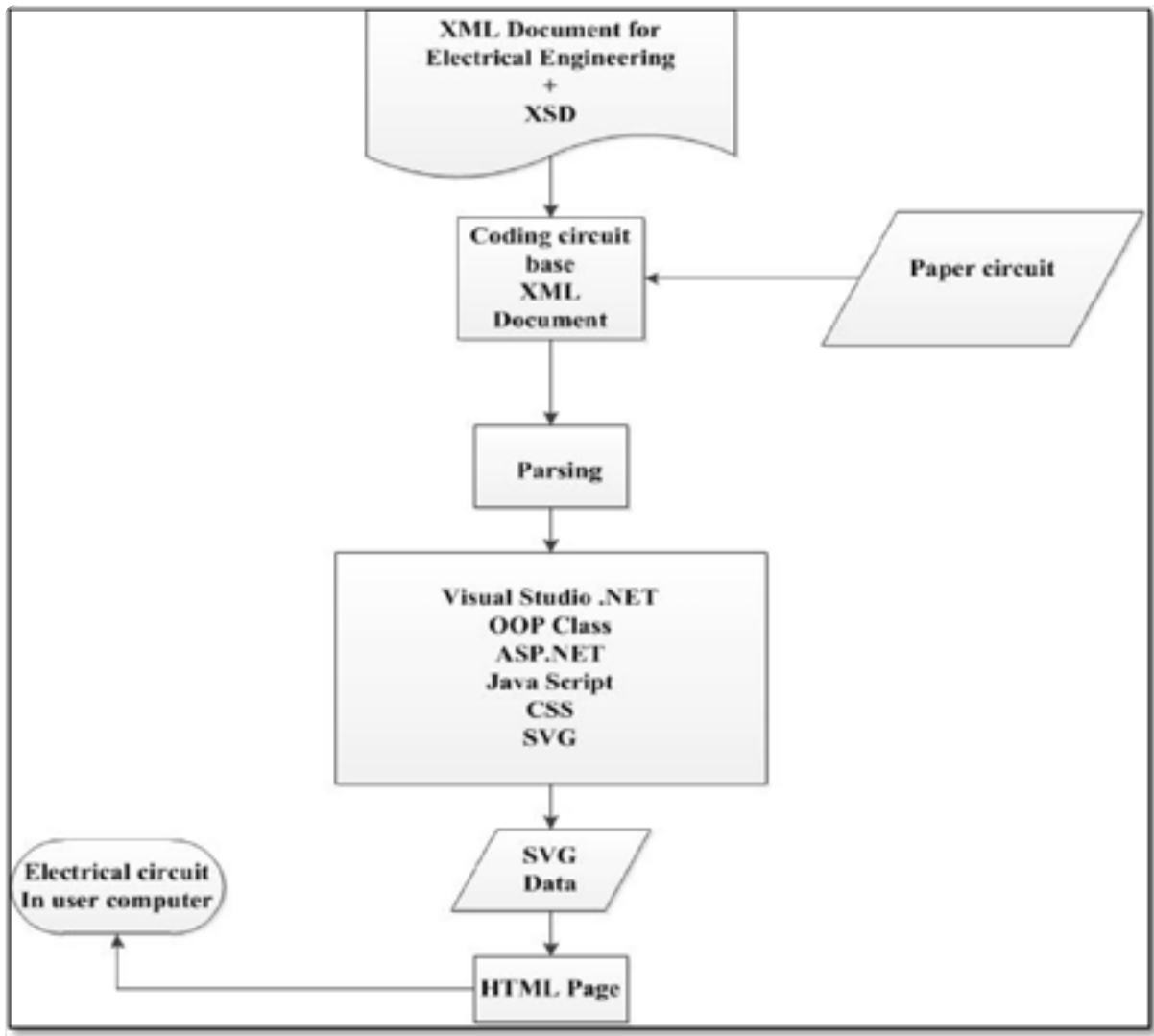
### Overview

Last chapters describe the importance of XML language and explain the advantages of using XML in many fields such as MathML, CML, WML.

This chapter presents the proposed model, to create a new markup language base XML. The new designed application is based on the new language. The proposed model can show how we can advance from the new language. The proposed model is a website designed to draw the electrical circuits. This section has a list of the syntax that should be cared for when creating a new language based XML markup language, and explaining the XML schema definition (XSD) for the markup language.

### 3.1 Proposed model

In the proposed model, the user wants to convert paper circuit to computer circuit using XML code. The first step is creating electrical engineering, language base XML. The user, then, enters a descriptive code based on electrical engineering language on the (text box) and press on (draw) button as mentioned in section (3.2). In the second step, the model validation form code by parser. After that, the model runs function and math equation to get SVG data, SVG is explained in section (3.3). Then (draw page) opens to get the output as electrical circuit. Figure (3.1) explains the proposed model steps.



**Figure (3.1): proposed model.**

Drawing circuit is done on SVG environment. Using SVG in drawing, it adds more advantages for the user as mentioned in chapter two. SVG is used to create vector graphics data over the web. SVG provides advantages more than current methods such as JPG, or GIF. Because vector graphics describe graphical information in terms of lines, curves not as pixel bitmap.

### 3.2 Creating a New Markup Language for Electrical Engineering.

To create the new markup languages for electrical engineering base XML, we should create XML document has the owner tags; XML language can create tags as needed. This chapter discusses how we can create this document and explain the XML schema definition (XSD) to validate documents.

#### 3.2.1 Classifications Tables of Electrical Symbols:

This section will represent the electrical symbol which will be used, and classifies these elements in tables. These elements were collected from many sites for electrical engineering. And represent the most elements in electrical, if any element is not found in table we can add it easily at any time. The following tables show the element we collected to create the language.













Symbol	Component name	Meaning
<i>Wire Symbols</i>		
	Electrical Wire	Conductor of electrical current
	Connected Wires	Connected crossing
	Not Connected Wires	Wires are not connected

Table (3.1): Wire Symbols, [www.scribd.com](http://www.scribd.com).






Symbol	Component name	Meaning
<i>Switch Symbols and Relay Symbols</i>		
	SPST Toggle Switch	Disconnects current when open
	SPDT Toggle Switch	Selects between two connections
	Pushbutton Switch (N.O)	Momentary switch - normally open
	Pushbutton Switch (N.C)	Momentary switch - normally closed
	DIP Switch	DIP switch is used for onboard configuration
	SPST Relay	Relay open / close connection by an electromagnet
	Jumper	Close connection by jumper insertion on pins.
	Solder Bridge	Solder to close connection

**Table (3.2): Switch Symbols and Relay Symbols.**

Symbol	Component name	Meaning
<i>Ground Symbols</i>		
	Earth Ground	Used for zero potential reference and electrical shock protection.









**Table (3.3): Ground Symbols**

Symbol	Component name	Meaning
<i>Resistor Symbols</i>		
	Resistor (IEEE)	Resistor reduces the current flow.
	Resistor (IEC)	
	Photo resistor / Light dependent resistor (LDR)	Photo-resistor - change resistance with light intensity change.

**Table (3.4): Resistor Symbols.**

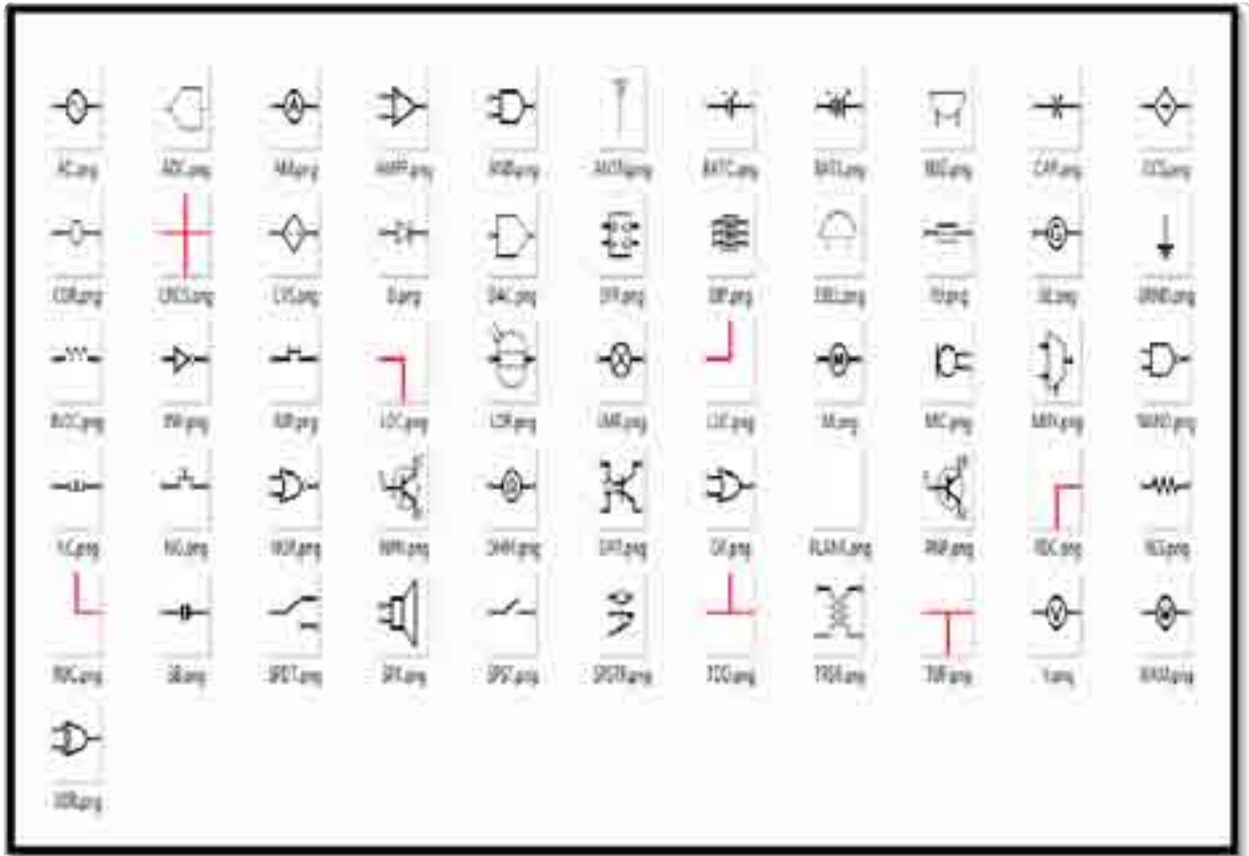
To see more element in Appendix A.

The electrical engineers need logical gate symbols in an electrical circuit. So, in this model symbols are added in XML documents and also tags are added to it. The next table (3.5) shows logic gate symbols.

Symbol	Component name	Meaning
<i>Logic Gates Symbols</i>		
	NOT Gate (Inverter)	Outputs 1 when input is 0
	AND Gate	Outputs 1 when both inputs are 1.
	NAND Gate	Outputs 0 when both inputs are 1. (NOT + AND)
	OR Gate	Outputs 1 when any input is 1.
	NOR Gate	Outputs 0 when any input is 1. (NOT - OR)
	XOR Gate	Outputs 1 when inputs are different. (Exclusive OR)
	D Flip-Flop	Stores one bit of data.
	Multiplexer / Mux	

**Table (3.5): Logic Gates Symbols.**

All images we need to draw in the electrical circuit are collected in a table; and all images will be given the same size (48\*48) pixels to be standard. figure (3.2) shows the images collected to be used in the proposed model.



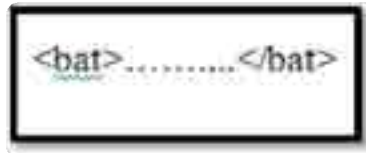
**Figure (3.2):** images used in the proposed model.

### 3.2.3 Syntax and the XML Rules:

The new language derived from XML language. To create this language, all conditions and rules in XML language should apply with care about the XML syntax. (Grijzenhout, S, 2013)

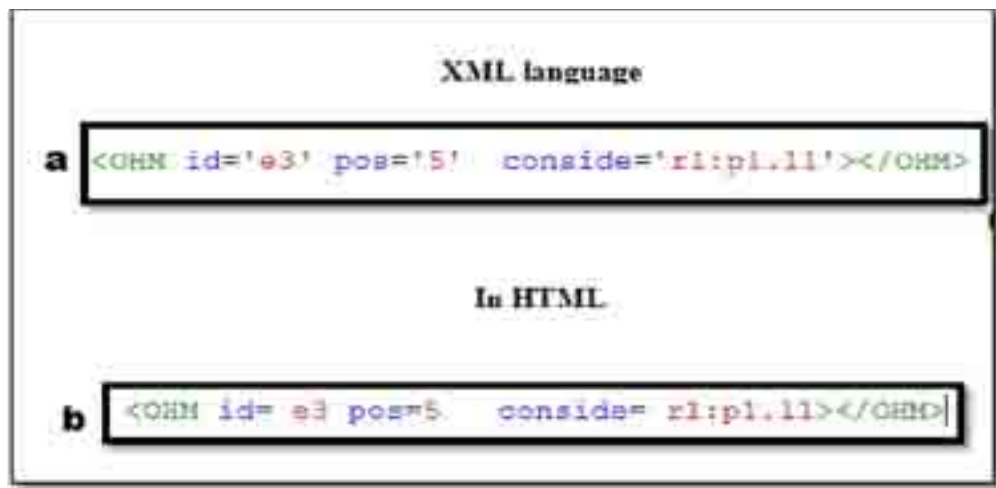
In XML the document should be well formed and should be valid. XML document should have to follow all rules in XML to be well formed. This rule is explained in the following section as w3schools.com.

- All XML Elements Must have open and Closed Tag, figure (3.3) description tag for element name (bat) opened and closed.



**Figure (3.3): (bat) element has an open and close tag.**

- The attribute values must be quoted. In XML tags ,the value must quote as shown in figure (3.4.a). While in html, the value doesn't quote as shown in figure (3.4.b). figure (3.4) shows the difference between html and XML when writing values for attribute.



**Figure (3.4): writes value in XML and HTML.**

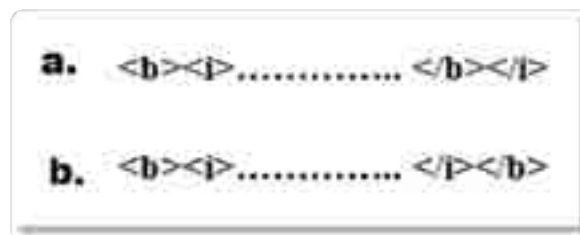
- XML tags are a sensitive case when writing open tag in capital letters, also closed tag should be written in capital letters. Figure (3.5) shows the error in the code because the element (OHM) is written with capital letter at open tag but small letters are used to write closed tag.



```
<OHM id= a1 pos=5 consid= r1:pl.11> </ohm>
```

**Figure (3.5): Error code for case sensitive.**

In XML, all elements must be properly nested within each other. Figure (3.6. a) shows nested tags. While Figure (3.6.b) shows properly nested tags.

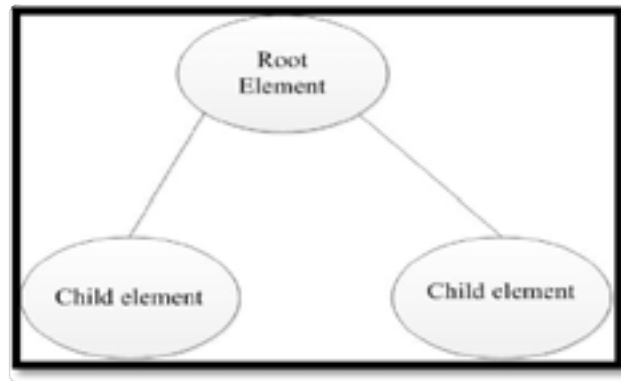


**a.** <b><i>..... </b></i>  
**b.** <b><i>..... </i><b>

**Figure (3.6): ordered tags in XML and HTML.**

- Documents in XML should have a root element.

XML documents should contain a single element that is the parent of all other elements. This element is called the root element. This root can have a sub element called child elements. Figure (3.7) shows root and child elements.



**Figure (3.7): Root and child element.**

- Comments in XML

The syntax for writing comments in XML is similar to HTML. The following example shows the comment in an XML document.

```
<!--<?xml version="1.0" encoding="utf-8" ?>-->
```

### **3.2.3 Writing the XML Codes.**

After collecting the elements, now we start to write tag in XML document; this document represents a new markup language for electrical engineering.

- **Declaration document.**

Any XML document should start with the declaration. Declarations provide information to the browser, the declaration has the language, declaration statement and version. figure (3.8) defines the XML version (1.0) and the encoding used (utf-8).



```
<!--<?xml version="1.0" encoding="utf-8" ?>-->
```

**Figure (3.8): Declaring the XML document.**

- **Create the root element:**

Any XML document should have the (Root) element. The root element is the parent for other elements and is the first element tag for XML file. In this document the root element is a (VOC) as shown in figure (3.9).



```
<!--<?xml version="1.0" encoding="utf-8" ?>-->
<VOC xmlns="http://tempuri.org/VOC.xsd">
```

**Figure (3.9): create the root element (VOC)**

- **Create child Elements:**

After creating the root element, it is also advisable to create the child elements. Child elements nest inside the root element. In figure (3.10) the child in this document is:

**< Elements >.**



```

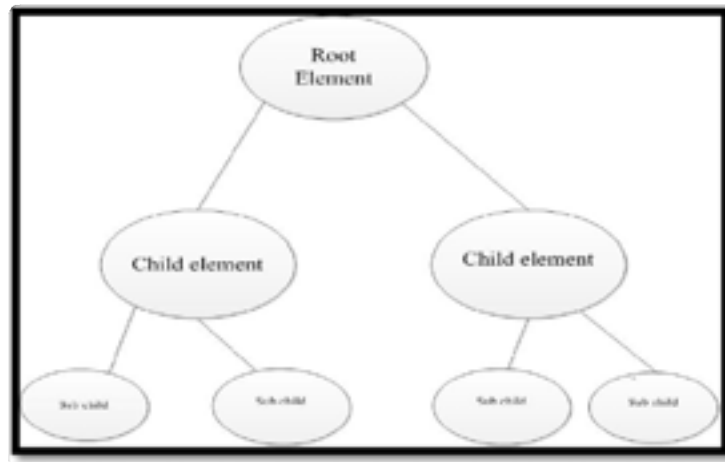
<!--<?xml version="1.0" encoding="utf-8" ?-->
<VOC xmlns="http://semur1.org/VOC.xsd">
  <Elements>
    <Tag>SP5T</Tag>
    <ElementName>Toggle Switch</ElementName>
    <url>SP5T</url>
    <Left1>0:23</Left1>
    <Left2>!</Left2>
    <Right1>48:23</Right1>
    <Right2>!</Right2>
  </Elements>
  <Elements>
    <Tag>NO</Tag>
    <ElementName>Pushbutton Switch</ElementName>
    <url>NO</url>
    <Left1>0:23</Left1>
    <Left2>!</Left2>
    <Right1>48:23</Right1>
    <Right2>!</Right2>
  </Elements>
</VOC>

```

Figure (3.10): creates child element <Elements>.

Once creating child element, nest Sub child elements inside of child elements to hold the data you want to store.

Such as (tag, Element name, left1, right1, left 2, right2). figure (3.11) explains the tree structure for XML document.



**Figure (3.11): Tree Structure of the XML Document**

XML Tree for a proposed model is shown in figure (3.12). The root element is vocabulary (VOC), the child element is (Element), and the sub child element is (tag, the element name, URL, left1, left2, right1, and right2).

Every element in this document should have the tag and element name and (URL) and connection point. These points represent connection of the element with other elements. Each element has four connection sides for every element (left1, right 1, left 2 and right 2)

After adding all elements in the document, then the root should be closed (VOC).

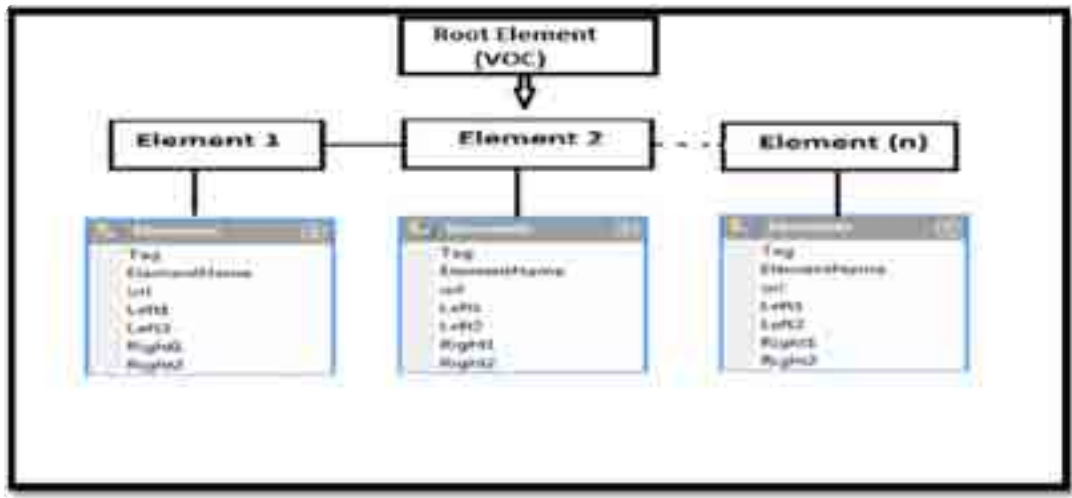


Figure (3.12): The XML Tree for the Proposed Model.

### 3.2.4 XML Parsers

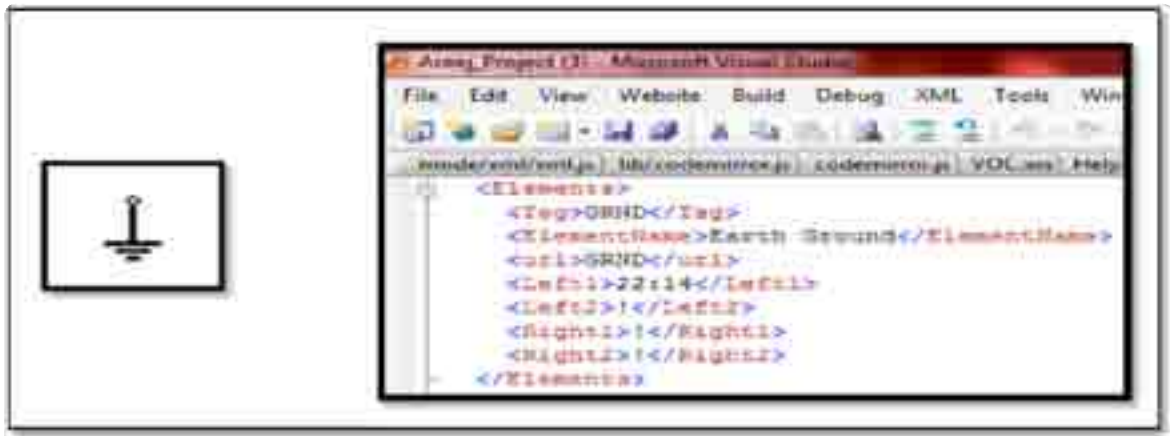
After the XML document is created, it needs to be validated by an XML parser. In our model the (.NET) Parser was used to validate the document. The parser aims to interpret the document's code and to verify that it satisfies all of the XML specification for document structure and syntax. (Chiu, 2006) and (PATRICK CAREY, 2003). "XML parsers are strict. If one tag is omitted or a character is lowercase when it should be uppercase, the parser will report an error and rejects the document. But that rigidity was built into XML to correct the flaw in HTML that gave web browsers too much discretion interpreting HTML code. The end result is that XML code accepted by the parser is certainly can apply the same everywhere."

### 3.2.5 Connecting the Elements together:

In the last section we collect the elements that are needed in the schema and electrical circuits ,and give it the standard size 48\*48 Pixels. To connect these elements with others four original points for each element to represent connection points were determined, every element has connection point left1, right1, left2, or right2.

Some elements need four connection points, but others need only one point. If any element doesn't need any connection point put in connection tag (!) Such as the following example `<Left2>! </Left2 >`, this means that the element doesn't need the left 2 connection points ;so put (!).

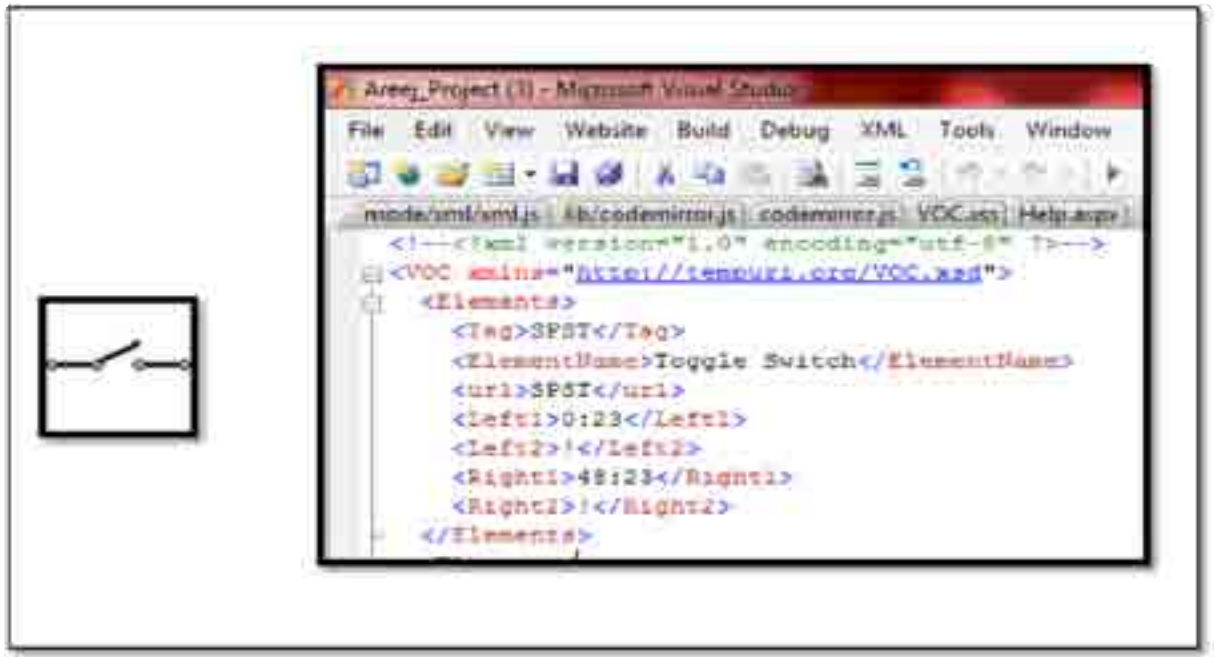
The figure (3.13) shows GRND element that needs one connection point; this point is left1; this element does not have left2, right1, right 2 connection side.



**Figure (3.13): The XML Code represents an Element that has One Connection Point**

figure (3.14) shows an element needing two connection points such as (SPST):

SPST element has two connection side left1 and right2.



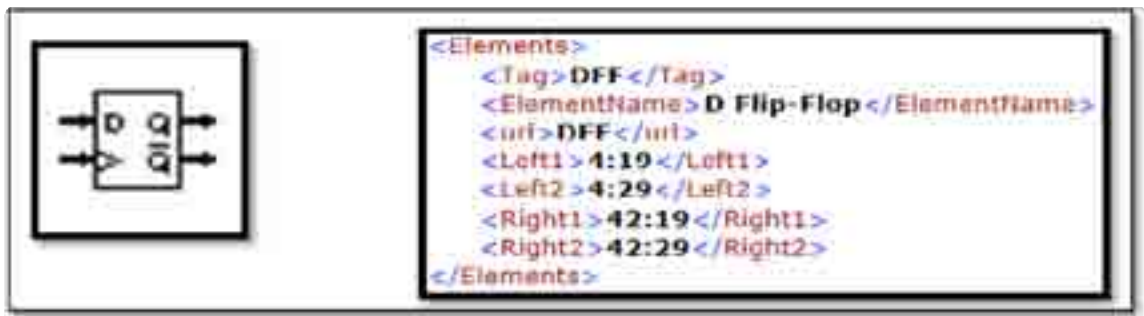
**Figure (3.14): XML code represents element has two connection points.**

The following figure (3.15) shows an example for element that needs three connection points such as NBN. NBN can connect with elements from three points.



**Figure (3.15): The XML Code represents an Element that has Three Connection Points**

figure (3.16) shows an example for element that needs four connection points such as DFF. DFF can connect with elements from four connection sides.



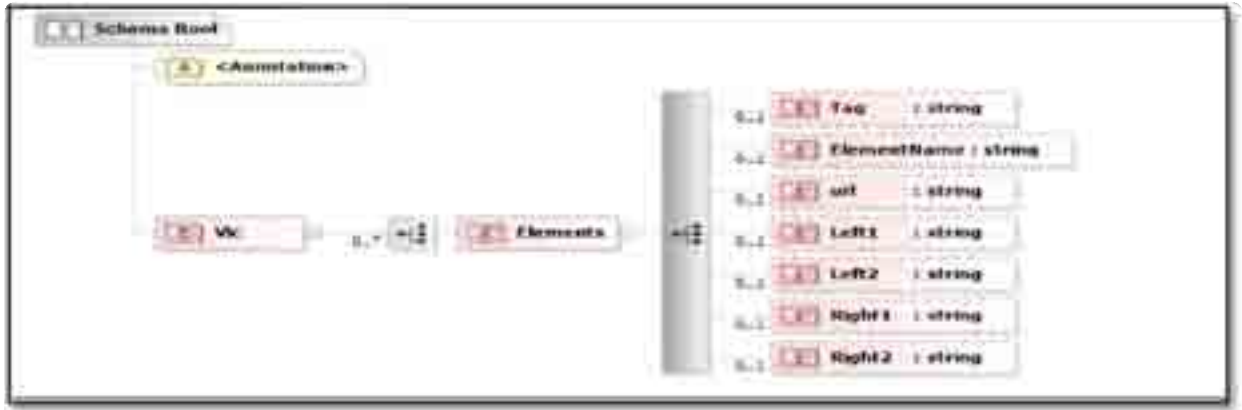
**Figure (3.16): XML code represents element that has four connection points.**

### 3.2.6 XML Schema definition (XSD):

(Fallside, 2004) XML scheme alternative data type definition (DTD) XML schema describes the structure of an XML document. XSD defines attributes and elements in a document and determines the child elements and the order of child elements. It defines data types for elements and attributes. (Frank Atanassowa, 2006) “an XML document is supplied along with a schema to a Schema processor, which parses and type-checks the document according to the declarations. This process is called validation”. We used XSD in our model because, “you could write a DTD for an XML page and accomplish some of the same goals. Because a schema is written in XML, there is no new syntax or rules to understand. If you can write a page in XML, you can write an XML schema.”

XML schema diagram contends from the Root element “VOC” and <Annotation> and <Elements>. The annotation element is a top level element that specifies schema comments, the comments serve as inline documentation. The <Elements> each element contains from: (Tag, Element Name, URL, left1, left2, right1, right2) as shown in figure (3.18).

Figure (3.17) shows the drawing by liquid XML program to represent XML schema diagram.



**Figure (3.17): The XML Schema Definition for Electrical Markup Language.**

Element Name	Element Type
Tag	string
ElementName	string
uri	string
Left1	string
Left2	string
Right1	string
Right2	string

**Figure (3.18): (XSD) Elements.**

The XML document is very flexible because one can add any element needed at any time without effecting the document. Having finished collecting all elements, we have the new markup language for electrical engineering. This new language based XML language, but it's customized for electrical engineering to draw our circuits. Figure (3.19) shows a data grid for the element. We can add any element or remove from this table.



The figure (3.19) shows data for elements which are ( Tag , Element Name , URL and connection side ).

The connections side ( left1, left 2 , right 1, right 2 ) are found manually for each element.

Data							
Data for Elements							
Tag	ElementName	url	Left1	Left2	Right1	Right2	
ANTN	Antenna / a	ANTN	22:39				
INV	NOT Gate (I	INV	0:23		48:23		
AND	AND Gate	AND	0:18	0:28	48:23		
NAND	NAND Gate	NAND	0:18	0:28	48:23		
OR	OR Gate	OR	0:18	0:28	48:23		
NOR	NOR Gate	NOR	0:18	0:28	48:23		
XOR	XOR Gate		(null)	0:28	48:23		
DFP	D Flip-Flop	DFP	4:19	4:29	42:19	42:29	
CROS	Cross Lines	CROS	0:23	24:0	24:48	48:23	
TUP	Tee Up	TUP	0:23		24:48	48:23	
TDO	Tee Down	TDO	0:23	24:0		48:23	
LDC	Left Down C	LDC	0:23		24:48		
RDC	Right Down	RDC	0:23	24:0			
LUC	Left Up Corn	LUC		24:0		48:23	
RUC	Right Up Co	RUC			24:48	48:23	
PLANK	PLANK	PLANK	48:23				
PLANK2	PLANK2	PLANK2	23:23				
PLANK3	PLANK3	PLANK3	0:23				

Figure (3.19): Data grid for XML document.

### **3.3. Designing the web application which supports the New Language.**

The Proposed module was developed to allow applications to transfer data between them. In the proposed model, the web site connected to XML document was designed. With this web application the electrical engineering can draw their schema and electrical circuit online without needing any software programs. All you need is a connection to the internet and an internet browser.

The last part of this chapter discusses how to create the customized markup language base XML language to support electrical engineering in their research and work. This section explains how designed web application is based on the markup language. This application helps to draw electrical circuits with more flexibility than other programs because this application base XML markup language.

Using XML in web application provides users with advantages and benefits. With this application, engineer can see how XML can provide many features when using it in our projects. Using XML language can add many advantages such as flexibility, storing data, accuracy, and exchanging information easily over the web in a small space in storage.

## **Tools and programming language.**

The proposed model is divided into two stages. The first creates a markup language for engineering by using XML syntax, XSD and web browser.

The second stage designs a web site for drawing electrical circuit base electrical engineering, language by using visual studio net 2005 (visual basic.Net, ASP.NET) and using HTML language, Java script, and SVG (scalable vector graphic), CSS (Cascading Style Sheet), Code mirror for style.

(Visual studio .NET 2005) (Visual basic.Net, ASP. NET) is used in the Client side for drawing circuit using SVG an XML document and Java script (Code mirror) and XML Parsing to Read a Simple XML File and Retrieve Values. (.NET parsing) used to read the circuit code to give it the meaning of full data for the browser.

### **3.3.1 Write the descriptive code in text box.**

The code must have the root element, this root is <C> for all circuits. Then, it calls the wanted element to draw by tag from XML document, and writes all attribute in this tag.

The first attribute determines the (id) for every element we need in a circuit such as e1, e2, e3. After determination that the position we want on screen. Then, it determines the connection side of the element. The following figure (3.20) explains the structure of code.

```

<C>
< Element id='string value' pos='number value' conside='number(1:4)'>
</Element>
<C>

```

**Figure (3.20): The Description of the Structure Code.**

**Where:**

1-(Id) represents the element ID.

2-(Pos) represents the element position on screen.

3-(Conside) represents a connection point from (my Side): to element want to connect with (Target Side).

4-Four (Conside) is: 1: first left side, 2: second left side, 3: first right side, 4: second right side {l1, r1, l2, r2} separated by ";".

5-(Rotate) if you want to rotate element 90° "clockwise".

**Determine the ID.**

The user can write any (id) needed in tag, but this id should be unique.

**Determine the position.**

After writing the id ,the user should give the position of elements. The positions are determined by dividing the screen into small squares; every square has 48 high and 48 width pixels because the shape of elements in XML document gives the standard size for

all element 48\*48 pixels. Screen splits based image size as Small Square every square has a number from (0 To (48\*48) -1).

Screen spilt 48\*48. Each square represents the position number as shown in figure (3.21).

0	1	2	3	.	.	46	47
48	49	50	51	.	.	94	95
96	97	98	99	.	.	142	143
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
2255	.	.	.	.	.	2302	2303

**Figure (3.21): Table represents the Numbers Position on the Screen.**

In html, the user should find (x, y) point of element, but that is not a flexible way in this model, So the user doesn't need to find (x, y) point for element, only gives the position for element base table in figure (3.21) such as (0, 1, 2, 3, 4,.... 2303). When the user writes the position in text code , the model finds (x, y) point for element automatically by equitation in figure (3.22).

```
p.X = (pos Mod 48) * 48
p.Y = Math.Floor(pos / 48) * 48
```

**Figure (3.22): Equitation to find (x, y) point automatically.**

This equation finds the point (X, Y) for an element when the user gives the position number.

If the user wants to draw the element in position (4) ; the equation gives the point by this equation = (240, 0).

The model collects this point automatically (X=240,Y=0). This equation gives more flexibility to determine position. Figure (3.23) shows (X,Y) point for position (4).

(240,0)							
0	1	2	3	4	.	.	47
48	49	50	51	52	.	.	95
96	97	98	99	100	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
2256	.	.	.	.	.	.	2203

**Figure (3.23): The (X,Y) Point is located in position four.**

(Olsson, September 26, 2008) In html the position is determined by three ways:

**Position: static.** This type means the element is not positioned and occurs where it normally would be in the document.

**Position: relative.** Y in this type use top or bottom, and left or right to move the relative element to where it would normally occur in the document. Such as down 30 pixels, and to the left 30 pixels.

**Fixed Positioning.** The element with fixed position is positioned relative to the browser window.

In this model the position is determined by the table way as a figure (3.21) and relative way.

### Determine connection side.

Consider in this model means the connection point from (my Side) to element wanting to connect with (Target Side) {l1,r1,l2,r2}. If there are more connection sides, separated by ";".

Figure (3.24) explains two elements, whereas every element has two connection sides left 1, right 1.

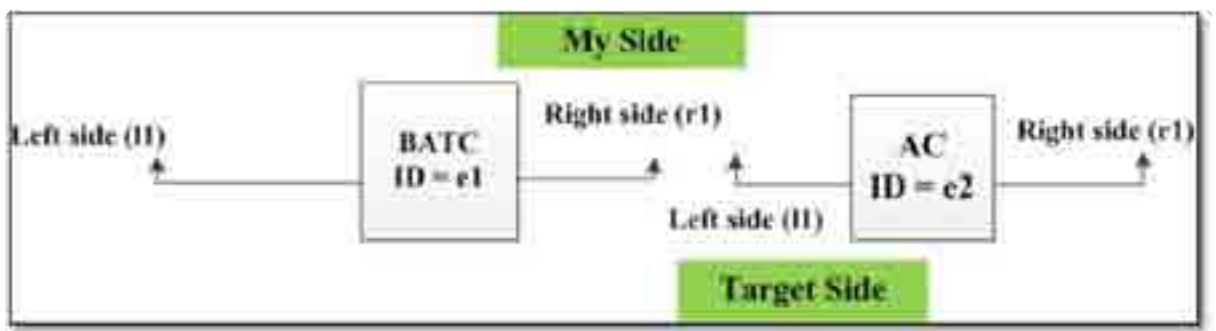


Figure: (3.24) Explain connection side.

```

1: <C>
2: <BATC id='e1' pos='1' conside='r1:e2.l1' > </BATC>
3: <AC id='e2' pos='3' conside='r1:e1.l1' > </AC>
4: </C>
5:

```

Figure: (3.25) Descriptive Code.

<BATC> in above example is `consider = 'r1: e2. l1; l1'`

"r1" is the first point from the element itself, e2. is a target element. l1 is the target side for element wanting to connect with it.

### Determine attributes rotate.

In equation the position determines a horizontal element, such as figure (3.26). But what happens if some circuits need vertical elements.



**Figure (3.26): circuit with horizontal elements only.**

To solve this problem another equation is created. This equation collects the point after rotating element  $90^\circ$  "clockwise".

first point collected by first equation in figure (3.22). For horizontal element.

A second equation in figure (3.27) should collect the new point after rotating elements as shown in figure (3.28) for vertical elements.

In two ways the user only writes the position, then the model collected the points automatically.



$$(x' - x_c) = \cos(90) \cdot (x - x_c) - \sin(90) \cdot (y - y_c)$$

$$(y' - y_c) = \sin(90) \cdot (x - x_c) + \cos(90) \cdot (y - y_c)$$

where

$X_c$ : x of center point.

$Y_c$ : y of center point.

Figure (3.27): An Equation is used to locate the Point after rotating the Element.

Figure (3.28) explains the point before rotating this point was found by first equation in figure (3.22). The second point after rotating is collected by the second equation in figure (3.28).

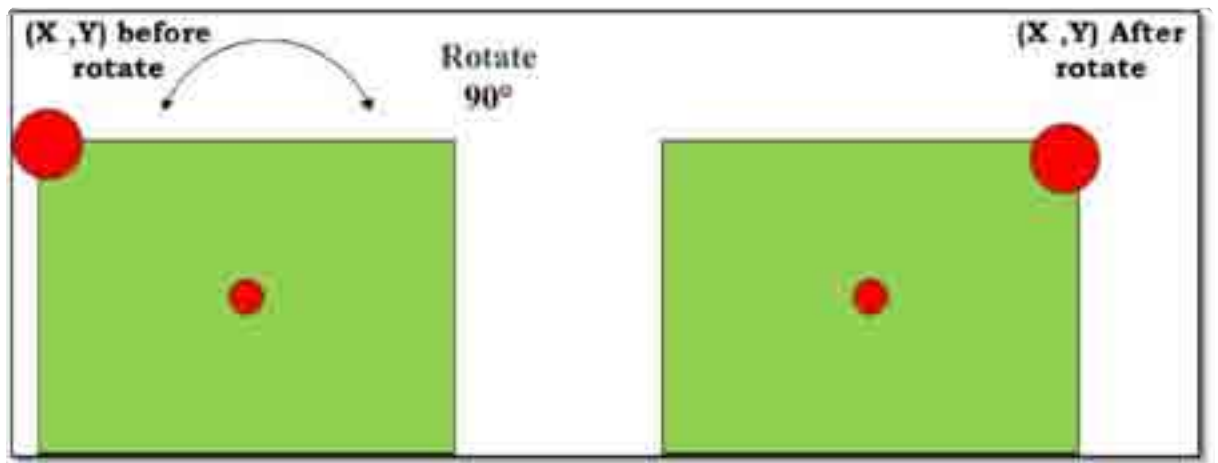


Figure (3.28): explain the point collected after rotating.

## Writing the Function and the Procedure.

### 1-Error functions.

This model aims to give more flexibility in use and discover the basic error. So this model cares about the error and it is designed with (check point function) to catch error and alert the user if there is an error in code.

### 2-Check node function.

The figure (3.29) shows (Check Node) function. This function checks if user has written the correct (id), and the position (pos). If the user doesn't write (id) the alerts message give (has no id), also If the user forgets the position value, the message gives (no position specification). If users do other error ,message gives "unspecified error".

```

Function CheckNode (ByVal x As XmlNode) As String
    Try
        If x.Attributes("id") Is Nothing Then
            Return x.Name & " : I have no id"
        End If
        If x.Attributes("pos") Is Nothing Then
            Return x.Name & " : " & x.Attributes("id").Value & " : no position specified "
        End If
        Return ""
    Catch ex As Exception
        Return "unspecified error"
    End Try
End Function

```

Figure (3.29): Check Node function.

Figure (3.30) is an example, where it shows user's screen that doesn't write the id for the element. The arrow refers to message alert to write (id). This message shows when user clicks on button draw.



**Figure (3.30): The Check Node Error.**

The next procedure in figure (3.31) is to load XML file in the text box and to give error message if the user doesn't write root tag <C> or close </C> tag.

In this procedure when user presses button, the page load XML file to call data from the document.

```

Processed Sub: Button1_Click(SyVal sender As Object, SyVal e As System.EventArgs) Handler: Button1_Click
Try
    Try
        XMLDoc.LoadXml(txtCode.Text)
    Catch ex As Exception
        lblErr.Text = "XML: Parsing Error: Document must start with <!DOCTYPE>"
    End Try
End Sub

```

Figure (3.31): procedure to load XML file and check root tag.

Figure (3.32) shows error for user forgetting the root that start with tag <C>.

```

1
2
3 <DIOD id='a3' pos='5' xolid='p1' consider='l1:p1.l1'></DIOD>
4 <DIAGR id='p1' pos='53' consider='l1:e4.r1'></DIAGR>
5 <DIAGR id='p2' pos='101' consider='l1:p1.l1'></DIAGR>
6 <AC id='a4' pos='49' consider='l1:e1.l1'></AC>
7 <IMP id='a5' pos='59' consider='r1:p2.l1'></IMP>
8 <BAPC id='q1' pos='97' consider='r1:e5.l1/l1:e4.l1'></BAPC>
9
10 </C>

```

XML: Parsing Error: Document must start with <C>

Figure (3.32): An Error has happened because of forgotten the Root tag by the User.

### 1- Check id function.

Function (Check Ids) checks if there are two elements have the same id, then alert error “there are two elements with the same id. Figure (3.33) shows (Check Ids) function. This function checks connection point and determine the connection points if is (left 1 or left 2 or write 1 or right 2). Every element has four cases of side, first is connected from

left 1 side, the second case is connected from left2, and the third case is connected from right 1 side or right 2 sides. This function checks this case and determines how many connection sides are for every element.

```

Function CheckRelishAndPoints(S,Val id As String, N,Val Cp As Integer) As String
Dim x As CNode = getElementBy(id, XMLDoc)
If x Is Nothing Then
Return "no such an element with id '" & id & "'"
Else
Select Case Cp
Case 1
If x.Left1.X = -1 Then
Return "there is no connection pint on first left side !"
End If
Exit Select
Case 2
If x.Left2.X = -1 Then
Return "there is no connection pint on second left side !"
End If
Exit Select
Case 3
If x.Right1.X = -1 Then
Return "there is no connections pint on first right side !"
End If
Exit Select
Case 4
If x.Right2.X = -1 Then
Return "there is no connection pint on second right side !"
End If
Exit Select
End Select
End If
Return ""

```

Figure (3.33): check id function.

### 3- Page load functions.

When the user writes the code, then presses on the (draw) button, the model checks errors, if there are no errors the model load draw page. Following the procedure in figure (3.34) used session to send data from the XML document in the text box.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    If Not IsPostBack Then
        If Session("XML") IsNot Nothing Then
            txtCode.Text = Session("XML")
        End If
    End If
End Sub
```

**Figure (3.34): page load by session.**

Next figure (3.35) shows code to send data from (SVG) to draw page and send XML data in text code. This procedure is used to show (SVG) data in draw page, and to call XML code in text box.

```
Dim str As String = checkIdc(ELI5)
If str <> "OK" Then
    lblerr.Text = str
    Exit Sub
Else
    Session("SVG") = drawC()
    Session("XML") = txtCode.Text
    Response.Redirect("drawpage.aspx")
End If
```

**Figure (3.35): send data from (SVG) to draw the page.**

## 4-Draw functions.

Figure (3.36) shows (Draw C) function. This function is used to draw electrical circuit.

In this function open the header tag to connect draw page with (SVG). We should open SVG tag and write code draw on SVG environment.

(If object <> “!” split “;”). In this line if connection side <> “!” this means the element has more than one connection sides and split between them by “;”.

The Draw function gets current sides and target side and connects between current side and the target side by lines.



```

Function drawC() As String
    Dim lines As New Queue
    Dim str As String = " <svg xmlns='http://www.w3.org/2000/svg' version='1.1' >"

    While Q.Count > 0
        Dim ELE As CLEObj = CType(Q.Dequeue, CLEObj)
        str += ELE.Shape
    End While
    For i As Integer = 0 To ELE.Length - 1
        Dim obj As CLEObj = ELE(i)

        If obj.ConnSide <> "!" Then
            If obj.ConnSide.Contains(";") Then
                Dim AllConnectionSides() As String = obj.ConnSide.Split(";")
                Dim trg() As Target = GetSides(AllConnectionSides)
                For j As Integer = 0 To trg.Length - 1
                    lines.Enqueue(GetLine(trg(j), obj))
                Next
            Else
                Dim TT As New Target
                Dim s() As String = obj.ConnSide.Split(";")
                TT.MySide = s(i)
                TT.TargetSId = getElementObj(s(i).Split(";")(0), XMLDoc)
                TT.TargetSide = s(i).Split(";")(1)
                lines.Enqueue(GetLine(TT, obj))
            End If
        End If
    Next
End Function

```

**Figure (3.36): draws circuit function.**

## 5-Function call shape

Figure (3.37) shows Function (shape). This function calls images from folder have all images. The image size standard (48\*48) pixels.

If the user wants the shape horizontal called it in normal way, but if the user needs the element vertical write attribute (rotate), then the code called the shape vertical.

When the user gives the attribute rotate true value the code takes the point for vertical element.



**Figure (3.37): calls shape function**

But if value = “!” the target gets all connections’ sides and draws line between (target) and (object).

That means the code collected all objects (shape) and then connects between all shapes of lines, then splits between my side and target element by “:” as shown in the next example.



**Example:**

**Consider = ' r1 : e5 . l1; l1: e4. l1 '**

### 6-Function calls line.

Figure (3.38) shows (Line) function to draw lines. This function determines the line size, color and connected point (x, y).

```

While lines.Count > 0
  Dim ll as line = CType(lines.Depose, line)
  str &= "(line id=" & ll.pl.X & " y=" & ll.pl.Y & " x2=" & ll.pl.X & " y2=" & ll.pl.Y & " style=" & strokeWidth & ",0,0) stroke-width:"
End While
str &= "</pre>
Return str

```

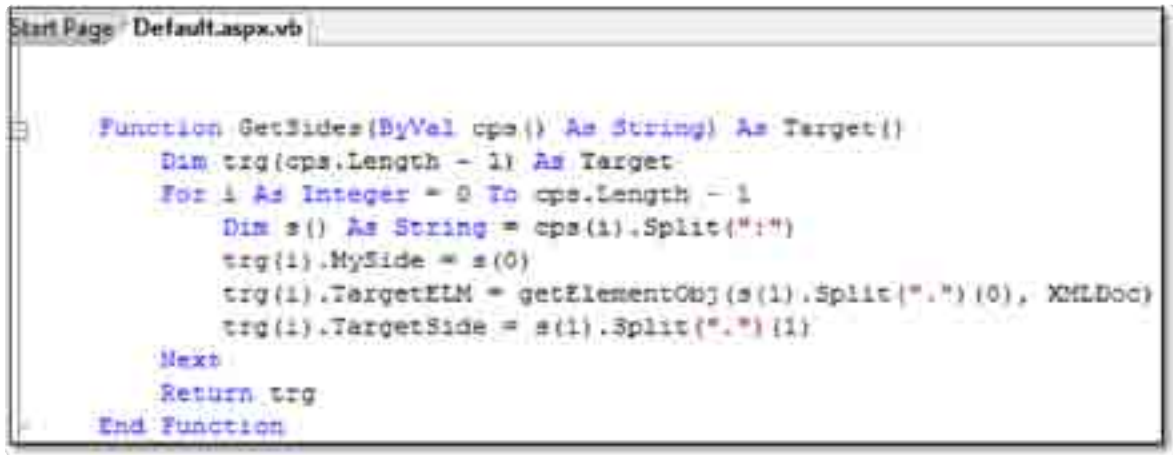
**Figure (3.38): function to draw the line.**

Function (draw) gets the wanted shape to draw. This function connects between shapes by determining point p1, p2 to connect it by line. p1 is my side, p2 is target side.

(Get line) Function determines if value p1 (my side) is left1 or left2, right1, right2. And value p2 (target side) is (left1 or left2, right1, right2). And Split between my side and target element by (:) and split between target element with target side by (.)

### 7-Get side function.

Figure (3.39) shows Get side function, this function gets current element side (my side), then get the element which wanted to connect with it (target element), and get the side for target element (target side).



```

Start Page Default.aspx.vb

Function GetSides(ByVal cps() As String) As Target()
    Dim trg(cps.Length - 1) As Target
    For i As Integer = 0 To cps.Length - 1
        Dim s() As String = cps(i).Split(":")
        trg(i).MySide = s(0)
        trg(i).TargetELM = getElementObj(s(1).Split(".") (0), XMLDoc)
        trg(i).TargetSide = s(1).Split(".") (1)
    Next
    Return trg
End Function

```

**Figure (3.39): Get Side function.**

## 8- Help page.

On the first page there are buttons to help the user, and their procedure is used to redirect the user to help page. In help page there are some notes for users to give information about how to write the code and how to use the website to draw their circuit. Figure (3.40) shows the help page designed to give the user some notes about how to write the code for their circuit.

Such as:

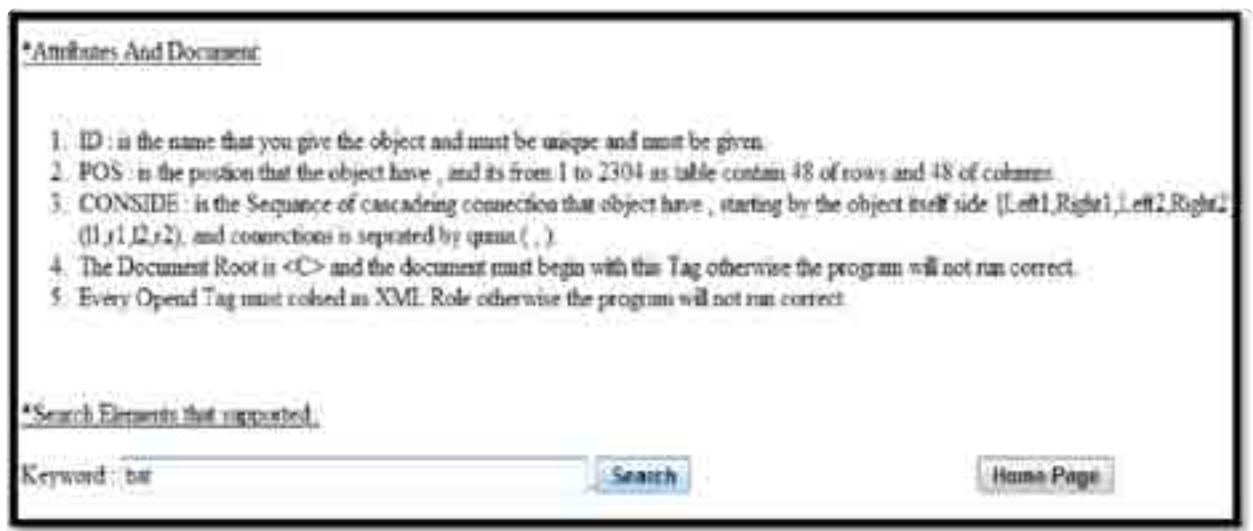
1. ID: is the name that you give the object and must be unique and must be given.
2. POS: is the position that the object has, and it's from 1 to 2304 as table contains 48 of rows and 48 of the columns.

CONSIDE: is the Sequence of cascading connection that objects have started with the object's side {Left1, Right1, Left2, Right2} ends with Target Object with its side {Left1, Right1, Left2, Right2}, Object. (l1, r1, l2, r2), and connections are separated by comma

(,).

3. The Document Root is <C> and the document must begin with this Tag otherwise the program will not run correct.

4. Every Opened Tag must close as XML Role otherwise the program will not run correct.



**Figure (3.40): help page.**

As shown in figure (3.41) in help page there are search buttons. The user enters any wanted word, then he presses search to show table which has all elements like words that you want. This table shows the element tag and element name, number of left connections and the number of right connection and element picture.

**\*Attributes And Document:**

1. ID : is the name that you give the object and must be unique and must be given
2. POS : is the position that the object have , and its from 1 to 2304 as table contain 48 of rows and 48 of column.
3. CONSIDER : is the Sequence of cascading connection that object have , starting by the object itself side (Left1,Right1,Left2,Right2) ends (M1,I1,I2,I3), and connections is separated by comma ( , )
4. The Document Root is <C> and the document must begin with this Tag otherwise the program will not run correct.
5. Every Opened Tag must closed as XML Role otherwise the program will not run correct.

**\*Search Elements that supported :**

Keyword:  Search [Home Page](#)

Element Tag	Element Name	Number Of Left Connections	Number Of Right Connections	Element Pic
BATC	Battery Cell	1	1	
BATS	Battery	1	1	







**Figure (3.41): Search page.**

The figure (3.42) shows search result for user's search about (bat) word. The result is shown in table has all information that user wants about the element.

In coding there are procedure, search button, search on XML schema and an element to find any word like the target word. The code filters all results that have same words by reading XML document and the element and search if there are any words like the words which user needs, then the result will be given in a tabular.

\*Search Elements that supported:

Keyword:

Element Tag	Element Name	Number Of Left Connections	Number Of Right Connections	Element Pic
AND	AND Gate	2	1	
INV	NOT Gate (Inverter)	1	1	
NAND	NAND Gate	2	1	
NOR	NOR Gate	2	1	
OR	OR Gate	2	1	
XOR	XOR Gate	2	1	

**Figure (3.42): result for search about (bat) word**

Figure (3.43) is a code to disable result search in table. This table has in the first column element Tag, second column has element's name third and fourth columns have a number of the left and right connection at final show to the picture of the word which user needs.

```

Dim dt As New Data.DataTable
dt.Columns.Add("Element Tag")
dt.Columns.Add("Element Name")
dt.Columns.Add("Number Of Left Connections")
dt.Columns.Add("Number Of Right Connections")

dt.Columns.Add("Element Pic")

For i As Integer = 0 To drs.Length - 1
    Dim dr As Data.DataRow = dt.NewRow
    Dim obj As New ClsObj(drs(i) (0))
    dr(0) = obj.Tag
    dr(1) = drs(i) (1)
    dr(2) = obj.NumberOfLeftEdges
    dr(3) = obj.NumberOfRightEdges
    dr(4) = ""
    dt.Rows.Add(dr)
Next

GridView1.DataSource = dt
GridView1.DataBind()

```

**Figure (3.43): code to order the result search in table.**

As seen in figure (3.43) the cell (4) is empty (“”). Because this cell for the image so left empty, figure (3.44) shows a procedure to call an image from source saved on “ELEMENTSPICS”.

```

Protected Sub GridView1_OnDataBound(ByVal sender As Object, ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs)
    If e.Row.RowType = DataControlRowType.DataRow Then
        e.Row.Cells(4).Text = "img src='ELEMENTSPICS' & e.Row.Cells(2).Text & ".jpg"
    End If
End Sub

```

**Figure (3.44): procedure to get picture in cell (4).**

The following image (3.45) shows Number of Left and Right function. This function is used to determine the number of left edges or right edge for elements.

```

Public Function NumberOfLeftEdges() As Integer
    Dim edg As Integer = 0
    If Me.Left1.X <> -1 Then
        edg += 1
    End If
    If Me.Left2.X <> -1 Then
        edg += 1
    End If
    Return edg
End Function

Public Function NumberOfRightEdges() As Integer
    Dim edg As Integer = 0
    If Me.Right1.X <> -1 Then
        edg += 1
    End If
    If Me.Right2.Y <> -1 Then
        edg += 1
    End If
    Return edg
End Function

```

**Figure (3.45):** function to find number the left and right edges for every element.

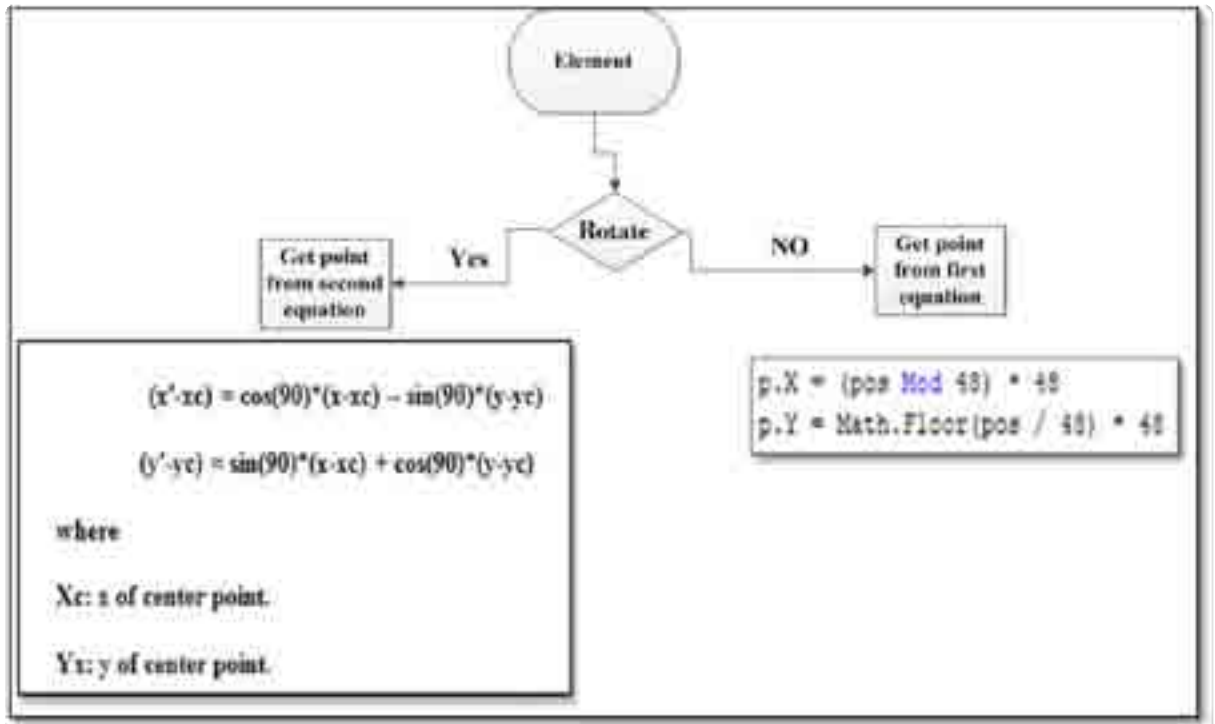
figure (3.46) shows rotate function this code has two cases: first one if the element has rotate value (true) the code rotate element  $90^\circ$  and get points from the equation was getting the new points after rotate. Second case if the element was horizontal the code would get point form first equation to get horizontal points.

```

If x.Attributes("rotate") IsNot Nothing Then
    Q.Enqueue(New TlsObj)(x.Attributes("L1").Value, x.Attributes("p1").Value, x.Base, x.Attributes("rotate").Value, x.Attributes("rotate"))
Else
    Q.Enqueue(New TlsObj)(x.Attributes("L1").Value, x.Attributes("p1").Value, x.Base, x.Attributes("rotate").Value, Nothing)
End If

```

**Figure (3.46):** rotates element



**Figure (3.47): Explain rotates element.**

The previously codes and functions were created by (visual studio. Net). The drawing does in (SVG) environment. The following (3.48) procedure is used to send SVG data to a text box in HTML page.

```

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    CC.InnerHtml = Session("SVG")
End Sub
  
```

**Figure (3.48): send (SVG) data in text box cc.**

In HTML page, we should write head and body. In this page can be used cascading style sheet to give style for page this code which is designed with code mirror. The code



mirror is called in default page to add style to code this style, to give the code a color. The following script is called code mirror in (text Code).

```
<script language="javascript" type="text/javascript">
var editor = CodeMirror.fromTextArea(document.getElementById("txtCode"), {
  lineNumbers: 20,
  mode: "text/html",
  matchBrackets: true
});
</script>
</body>
</html>
```

**Figure (3.49): script to call code mirror.**

This Code is designed to style page ([code mirror.net/](http://code-mirror.net/)). “Code Mirror is a versatile text editor implemented in Java Script in the browser. It is specialized for editing code, comes with a number of language modes and add-ons that implement more advanced editing functions. CSS is available for customizing Code Mirror to fit your application and extending it with new functionality.

Code Mirror is an open-source project shared under an MIT license. It is the editor which is used in Light Table, Adobe Brackets, Google Apps Script, Bit bucket and many other projects.

Code Mirror is “a code-editor component that can be embedded in Web pages. The core library provides only the editor component, no accompanying buttons and Code Mirror works with language-specific modes. Modes are JavaScript programs that help color (and

optionally indent) text written in a given language. So, in this thesis we used the code mirror with XML to give the project more functionality”.

## Main screen.

This web site has two pages; the first one is to write the markup descriptive code. The second page is to show a drawing of electrical circuit and has two buttons the first one is to draw the circuit, the second button is directed to help and search page. On first page there is a text box. The user writes the descriptive code base a new markup language as input, and then presses draw to get electrical circuit. After pressing the draw button the model will check if there is no error in descriptive code before the electrical circuit will appear. figure (3.50) shows the main screen in web site has a code written by the user in the text box:



```

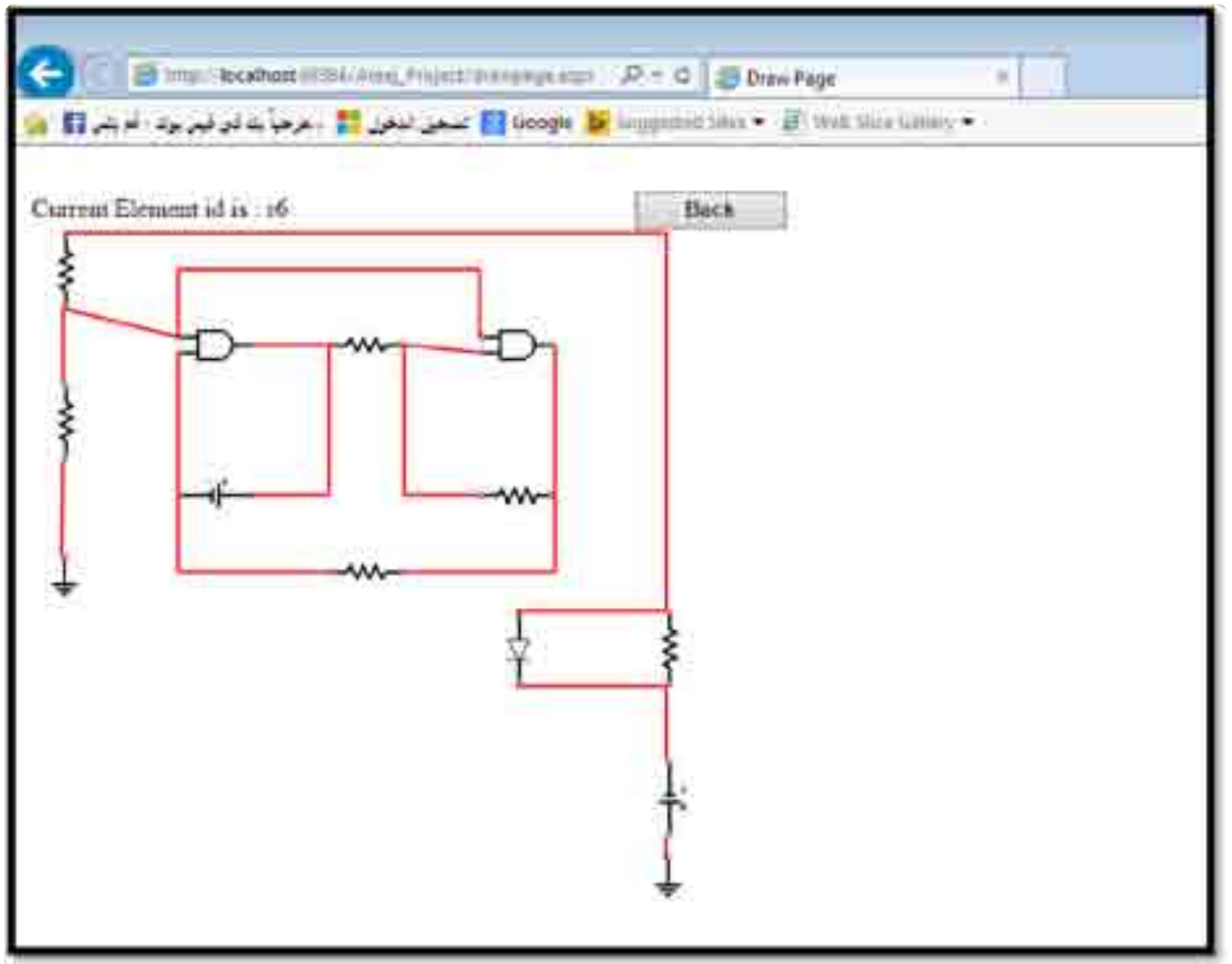
1 <<>
2
3 <RES id='r1' pos='0' consid='r1:r2.ll/r1:a1.ll' rotate='true' > </RES>
4 <RES id='r2' pos='36' consid='r1:r1.ll' rotate='true' > </RES>
5 <PLANK id='p6' pos='1' consid='l1:a1.ll' > </PLANK>
6 <AND id='a1' pos='50' consid='c1:r5.ll' > </AND>
7 <RES id='r3' pos='52' consid='r1:a2.ll' > </RES>
8 <PLANK id='p5' pos='8' consid='l1:r1.ll/l1:r4.ll' rotate='true' > </PLANK>
9 <PLANK id='p7' pos='5' consid='l1:p5.ll/l1:r4.ll' > </PLANK>
10 <AND id='a1' pos='56' consid='!' > </AND>
11 <BACC id='b1' pos='146' consid='l1:a1.ll'></BACC>
12 <RES id='r5' pos='150' consid='r1:a2.r1'></RES>
13 <PLANK id='p1' pos='142' consid='l1:r3.ll/l1:r3.r1' > </PLANK>
14 <PLANK id='p2' pos='147' consid='l1:b1.r1:r1:r3.ll' > </PLANK>
15 <PLANK id='p3' pos='151' consid='l1:r4.ll/l1:b1.ll' > </PLANK>
16 <RES id='r4' pos='146' consid='!' > </RES>
17 <PLANK id='p4' pos='132' consid='l1:r4.r1/l1:r3.r1' > </PLANK>
18
19 <CHRD id='g1' pos='122' consid='!'></CHRD>
20 <D id='d1' pos='146' consid='!' rotate='true'></D>

```

Draw Help

Figure (3.50) main screen has code written by users.

Now the user has finished the descriptive code, then presses the button (draw) to show the electrical circuit in figure (3.51).



**Figure (3.51): electrical circuit in draw page.**

In this page when user puts the mouse in any element the note tells him which element this is. That gives more flexibility to modify any error. In this figure the mouse move over element have (r6) id.

This page also have back button to back home page

When drawing circuit facing the problem online, because the lines cross with each other in a corner as shows in figure (3.52).



**Figure (3.52): Cross line problem.**

To solve this problem think about add attribute for element in the XML document. This element is Left down element, right down element, left, up corner, right up corner, top, or down. In this solution the user can fix this problem by determining the position by (x, y) point. figure (3.53) shows added attribute in the XML document.

```

<Elements>
  <Tag>TDO</Tag>
  <ElementName>Tee Down</ElementName>
  <url>TDO</url>
  <Left1>0:23</Left1>
  <Left2>24:0</Left2>
  <Right1>1</Right1>
  <Right2>48:23</Right2>
</Elements>
<Elements>
  <Tag>LDC</Tag>
  <ElementName>Left Down Corner</ElementName>
  <url>LDC</url>
  <Left1>0:23</Left1>
  <Left2>1</Left2>
  <Right1>24:48</Right1>
  <Right2>1</Right2>
</Elements>
<Elements>
  <Tag>RDC</Tag>
  <ElementName>Right Down Corner</ElementName>
  <url>RDC</url>
  <Left1>0:23</Left1>
  <Left2>24:0</Left2>
  <Right1>1</Right1>
  <Right2>1</Right2>
</Elements>

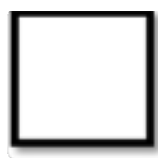
```

**Figure (3.53): added element to solve problem.**

This solution solves the problem, but after applying it. We see that it is not a flexible way for the user; this model should give more flexibility rather than other programs so try to get another solution more flexibility and ease.

This solution is done by adding another XML document element we called it a (PLANK). This plank as shown in figure (3.54) represents empty image (Wight image). (PLANK) is an image that doesn't appear when using it, so put (PLANK) in corner to connect it with element, this plank solves the cross line problem and gives more flexibility rather than the first solution.

**Figure (3.54): (PLANK) image.**



## Chapter Four

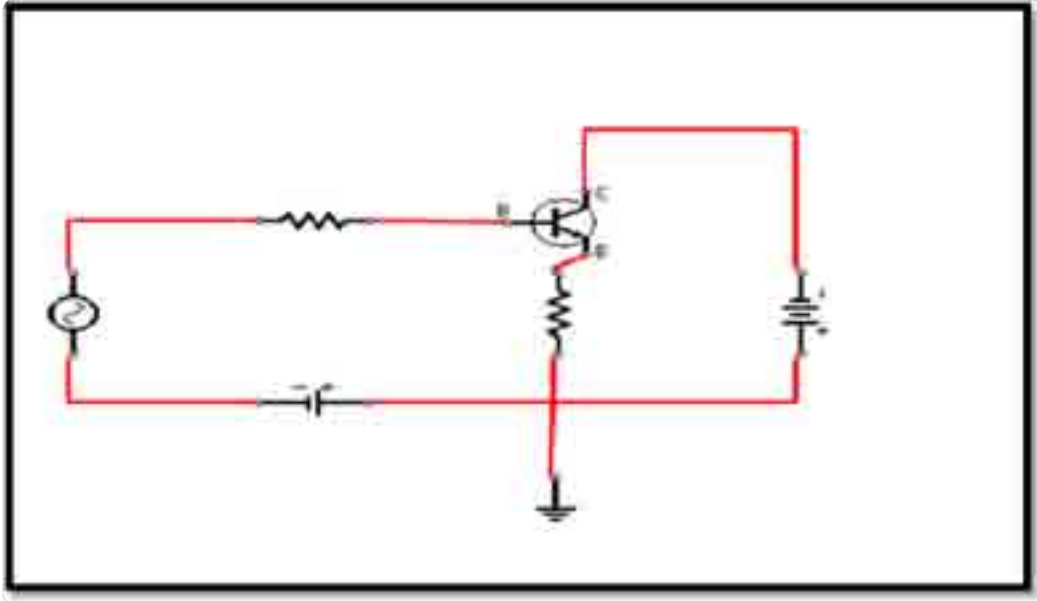
### EXPERIMENTAL RESULTS

#### 4.1 Comparisons and results.

Last chapter, we dealt with design and web application to draw electrical circuit, In this model, the drawing base markup language, the user writes an XML markup code and converts code to an electrical circuit. This circuit is displayed as XML file (text) not as an image. Many researchers are directed to use markup language in their projects to add more flexibility and advantages to their projects.

This chapter will present the comparison between the proposed model and the drawing software programs. The software programs draw electrical circuit by dragging the element of work space, after connecting elements with lines and saving images. All circuits in these programs are saved as an image, this image represents bitmap pixels. With proposed model the user writes a simple XML code and presses button (draw) to convert code to an electrical circuit. This circuit is saved as text (XML file), so the size file with new model has smaller size in storage rather than other programs, the proposed model isn't affected by resolution of screen. The following tables show a comparison between the proposed models with other three software programs.

All comparisons in this chapter are based on the same circuit shown in figure (4.1). This circuit is drawn in three programs and with the proposed model.



**Figure (4.1): basic circuit drawn with all programs.**

The comparison between (Express PCB), (Circuit Maker 2000) and (SEE Electrical LT) with (Electrical Circuit draws base XML) on hand:

- 1- File type.
- 2- Image size.
- 3- Size on disk.
- 4- Requirements.
- 5- Plate form independent.
- 6- Edit on circuit.
- 7- Extract information.
- 8- Media type.
- 9- Accuracy with zoom.
- 10- Sharing and exchange information.

### 1- File type and image size.

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>File of type</b>	<b>Bitmap image (. bmp)</b>	<b>WMF File (. wmf)</b>	<b>EMF File (. emf)</b>	<b>XML File (. XML)</b>
<b>Image size</b>	<b>96.3 KB (98,304, bytes)</b>	<b>8.06 KB (8,254 bytes) 10</b>	<b>2.63 MB (2,765,080 bytes) 2411</b>	<b>870 bytes (870 bytes)</b>

**Table (4.1): compares the base size image.**

As shown in table (4.1) there is a significant difference between models on handling image sizes. In Express PCB program the size is (98,622 bytes) that's in circuit maker the size 8.06 KB (8,254 bytes) on SEE Electrical image and size is 2.63 MB (2,765,080 bytes).

In the first program the image size doubled (113) rather than proposed model base XML, also the image size in circuit maker double (10) times rather than a model. In the third program the size is doubled (2411) rather than a new model base XML. Briefly, using (XML) in drawing give engineer significant results on image size. Figure (4.2) explains the result by numbers.



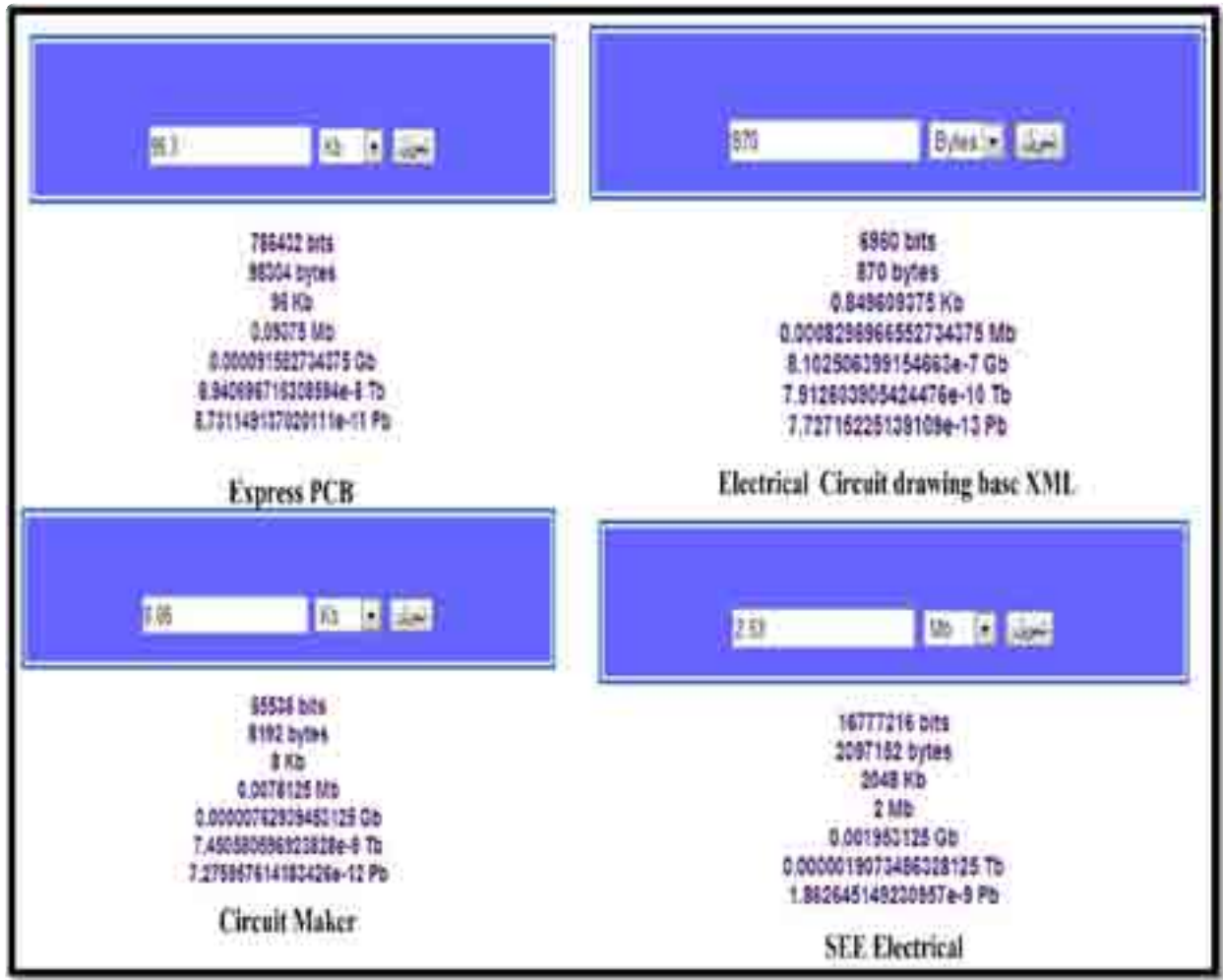


Figure (4.2): compares between software programs with proposed model base numbers.

Base on the significant result in image size the difference between sizes that mean the information sends faster and easier than other programs.

**2- Size image on disk.** The following table (4.2) shows the difference between sizes of the image on disk. Also the proposed model gets better results rather than another program.

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Size on disk</b>	<b>100 KB (102,400 bytes)</b>	<b>12.0 KB (12,288 bytes)</b>	<b>2.64 MB (2,768,896 bytes)</b>	<b>4.00 KB (4,096 bytes)</b>

**Table (4.2): Comparison base size image on disk.**

**3- Requirements.** The next comparison is done between model and programs base requirements, the user needed in drawing circuit. In all software programs we need to download software in large size on disk, but with new model base XML is needed only connection with internet and net browser as shown in the next table (4.3)

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Requirement</b>	<b>Download software program size 10.1 MB</b>	<b>Download software program size 35.1 MB</b>	<b>Download software program size 110.1 MB</b>	<b>Connection with internet and net browser</b>

**Table (4.3): Comparison based on the requirements.**

**4- Platform independency.**

Platform independency means Software can run on any platform such as Linux, Mac, UNIX and Windows and run on hardware platform such as PC and Mac. The software programs faces a problem when runs the program on any operating system. The new model is a platform independent.

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Platform independent</b>	<b>Not platform independent</b>	<b>Not platform independent</b>	<b>Not platform independent</b>	<b>Platform independent</b>

**Table (4.4): Comparison based on the plate form independent.**

#### **5- Edit on circuit.**

The proposed model features with flexibility to edit, with other software programs the user faces problems with editing their drawing, because it is saved as an image, if the user needs to edit any circuit on the web that requires, again, to draw circuit, but with new model the user can edit on circuit easily in any time without effects on circuit quality, because the circuit, is saved as a text file.

One of the most advantages of this application is that it is easier to modify without affecting other element in other software when the user saves images and sends it to their manager. If they have any element needed editing, they should again draw the image, but with new model the user can modify anything easily because editing is done on text not on the image.

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Edit on circuit</b>	Hard edit an image may be Again drawn	Hard edit an image may be Again drawn	Hard edit an image may be Again drawn	Easily modifiable because editing done on text not on the image

**Table (4.5): Comparison bases Edit on circuit.**

#### **6- Extract information from drawing.**

The output of drawing programs is image file so there are difficulties to extract or read information from the image, but with proposed model all information on the circuit can be extracted and read by inference engines, because the file type is text (XML file) as shows in the next table (4.6).

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Extract information from drawing</b>	Can't extract information from drawing	Can't extract information from drawing	Can't extract information from drawing	Can extract information from drawing

**Table (4.6): Comparison bases Extract information from drawing.**

#### **7- Media type.**

The next table (4.7) comparison between models of media type. In other software programs the media type is saved as bitmap pixels, but with proposed model the drawing in SVG environment.

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Media type</b>	<b>Bitmap pixels</b>	<b>Bitmap pixels</b>	<b>Bitmap pixels</b>	<b>Drawing in Vectore.SVG</b>

**Table (4.7): Comparison bases Media type.**

#### **8- Accuracy with Zoom.**

As mentioned above, the new model base XML is used to draw with SVG. With SVG the image scale that's meant can be zoomed without loss in quality and the drawing using SVG gives a clear image after printing it.

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Accuracy With Zoom</b>	<b>Accuracy affects with zoom for all circuits</b>	<b>Accuracy affects with zoom for all circuits</b>	<b>Accuracy affects with zoom for all circuits</b>	<b>The line Doesn't affect with zoom only shape affected with zoom</b>

**Table (4.8): Comparison bases Accuracy with Zoom.**

### 9- Sharing and exchange information.

The main function for XML files is to share and exchange information easily because the file is saved as a text file so sharing information over the web is more flexible rather than other programs as shows in the next table (4.9).

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>Sharing and exchange information</b>	<b>Not flexible</b>	<b>Not flexible</b>	<b>Not flexible</b>	<b>Flexible</b>

**Table (4.9): Comparison bases Sharing and exchange information.**

**Conclusion of comparisons on table (4.11):**

	<b>Express PCB</b>	<b>Circuit Maker 2000</b>	<b>SEE Electrical LT</b>	<b>Electrical Circuit drawing base XML</b>
<b>File of type</b>	<b>Bitmap image (. bmp)</b>	<b>WMF File (. wmf)</b>	<b>EMF File (. emf)</b>	<b>XML File (. XML)</b>
<b>Image size</b>	<b>96.3 KB (98,622 bytes)</b>	<b>8.06 KB (8,254 bytes)</b>	<b>2.63 MB (2,765,080 bytes)</b>	<b>870 bytes (870 bytes)</b>
<b>Size on disk</b>	<b>100 KB (102,400 bytes)</b>	<b>12.0 KB (12,288 bytes)</b>	<b>2.64 MB (2,768,896 bytes)</b>	<b>4.00 KB (4,096 bytes)</b>
<b>Requirement</b>	<b>Download software program size 10.1 MB</b>	<b>Download software program size 35.1 MB</b>	<b>Download software program size 110.1 MB</b>	<b>Connection with internet and internet browser</b>
<b>Platform independent</b>	<b>Not platform independent</b>	<b>Not platform independent</b>	<b>Not platform independent</b>	<b>Platform independent</b>
<b>Edit on circuit</b>	<b>Hard edit an image may be Again drawn</b>	<b>Hard edit an image may be Again drawn</b>	<b>Hard edit an image may be Again drawn</b>	<b>Easily modifiable because editing done on text not on the image</b>
<b>Extract information from drawing</b>	<b>Can't extract information from drawing</b>	<b>Can't extract information from drawing</b>	<b>Can't extract information from drawing</b>	<b>Can extract information from drawing</b>
<b>Media type</b>	<b>Bitmap pixels</b>	<b>Bitmap pixels</b>	<b>Bitmap pixels</b>	<b>Drawing in Vectore.SVG</b>
<b>Accuracy With Zoom</b>	<b>Accuracy affect with zoom for all circuits</b>	<b>Accuracy affect with zoom for all circuits</b>	<b>Accuracy affect with zoom for all circuits</b>	<b>The line Doesn't affect with zoom, only the shape affect</b>
<b>Sharing and exchange information</b>	<b>Not flexible</b>	<b>Not flexible</b>	<b>Not flexible</b>	<b>flexible</b>

Table (4.10): Comparison table.

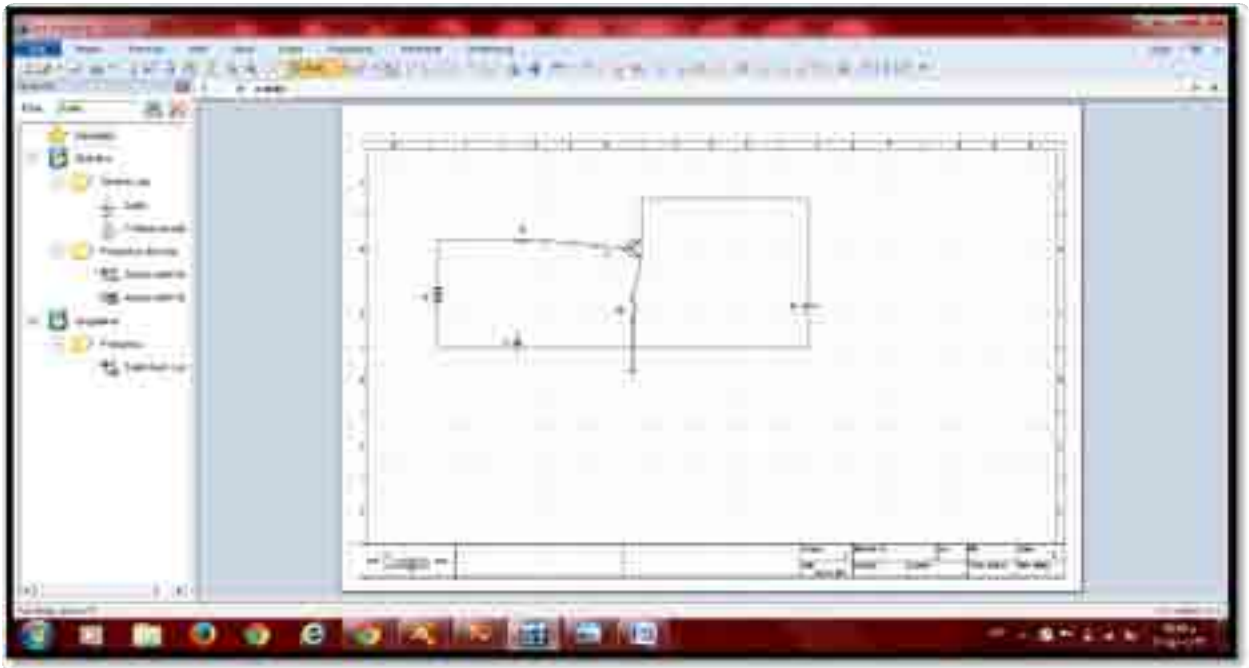
#### 4.2 Draw the same circuit on software program and proposed model.

Figure (4.3) shows the First program (Express PCB) interface.



**Figure (4.3): Express PCB program.**

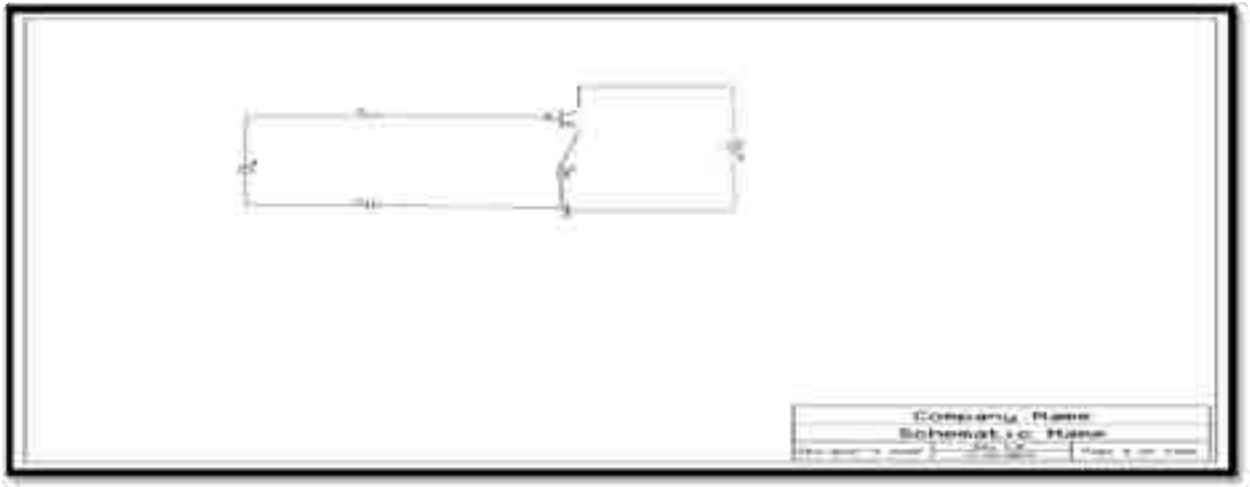
Figure (5.4) is for Second program Circuit Maker 2000 interface:



**Figure (4.5): User interface for Circuit Maker 2000.**

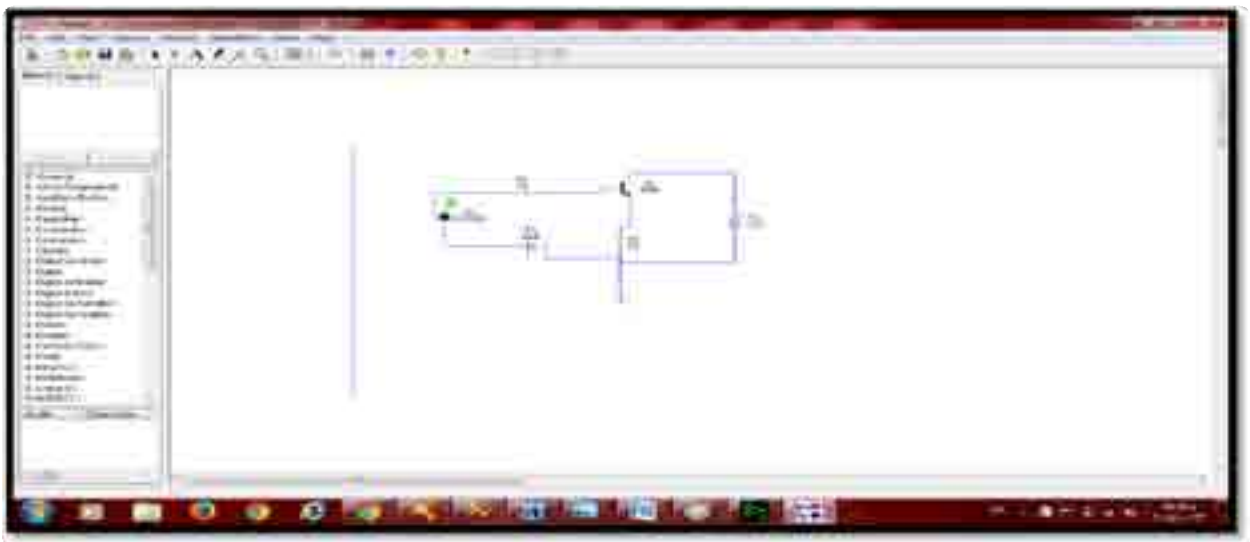
Figure (4.6) for circuit drawn by this program.





**Figure (4.6): Electrical circuit drawn by Circuit Maker 2000.**

Third program is SEE Electrical LT:



**Figure (4.7): User interface for SEE Electrical LT.**

Figure (4.8) is for electrical circuit drawn by (SEE Electrical LT).

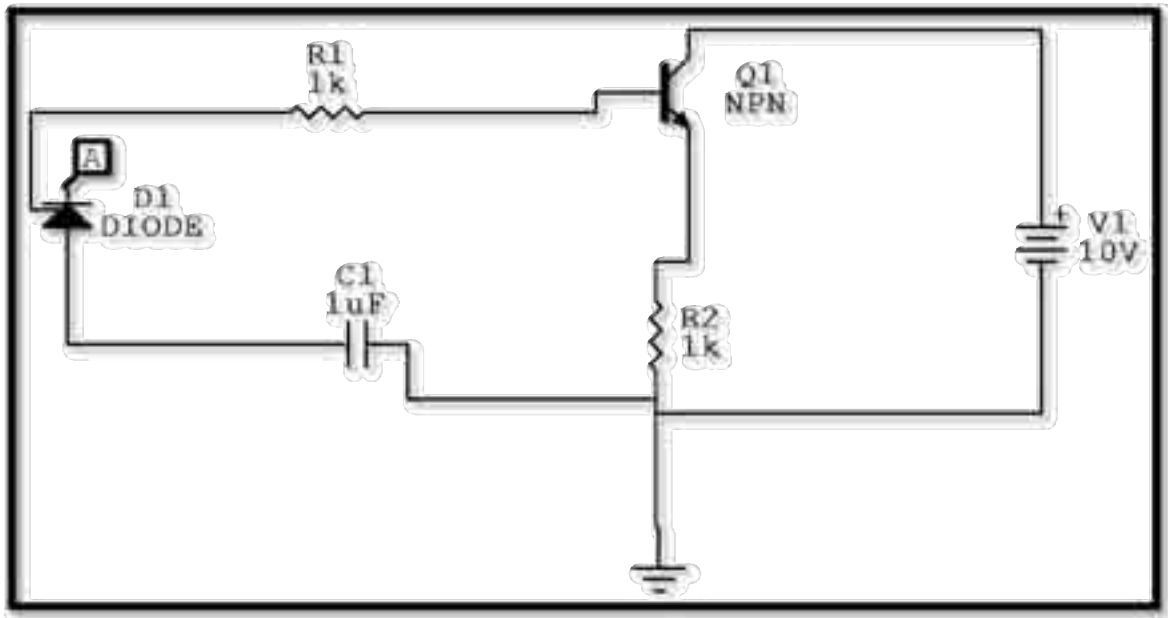
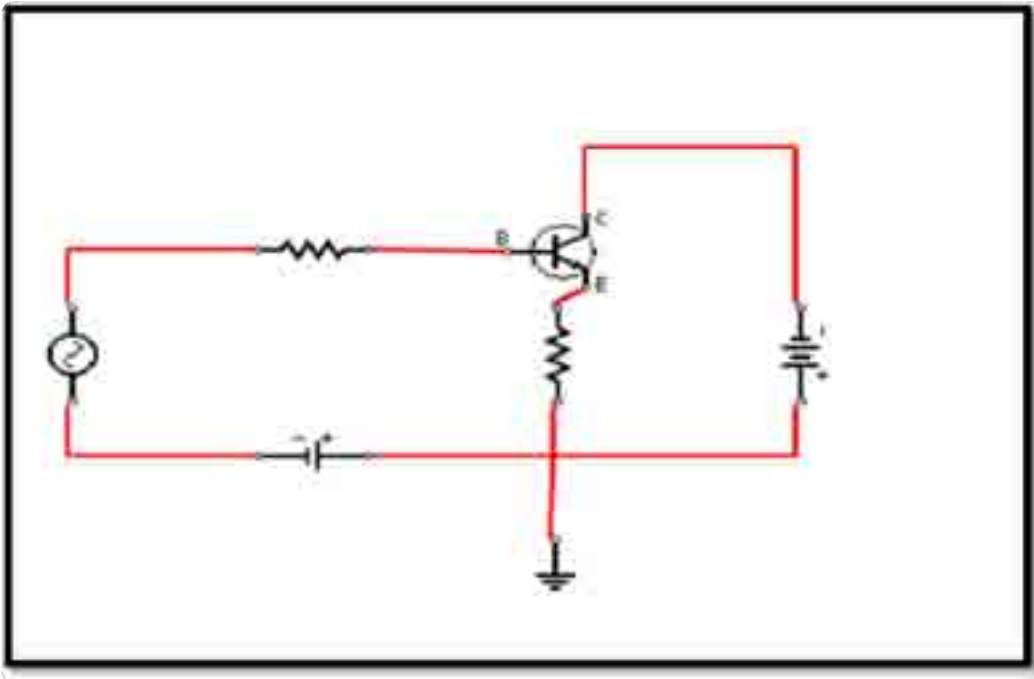


Figure (4.8): Electrical circuit drawn by (SEE Electrical LT).

Finally figure (4.9) shows the circuit that designed with a new model which was designed base in XML language.



**Figure (4.9): Electrical circuit design with our project.**

Example: the next figure explains the descriptive code written by users and shows the electrical circuit in figure (4.10).

```

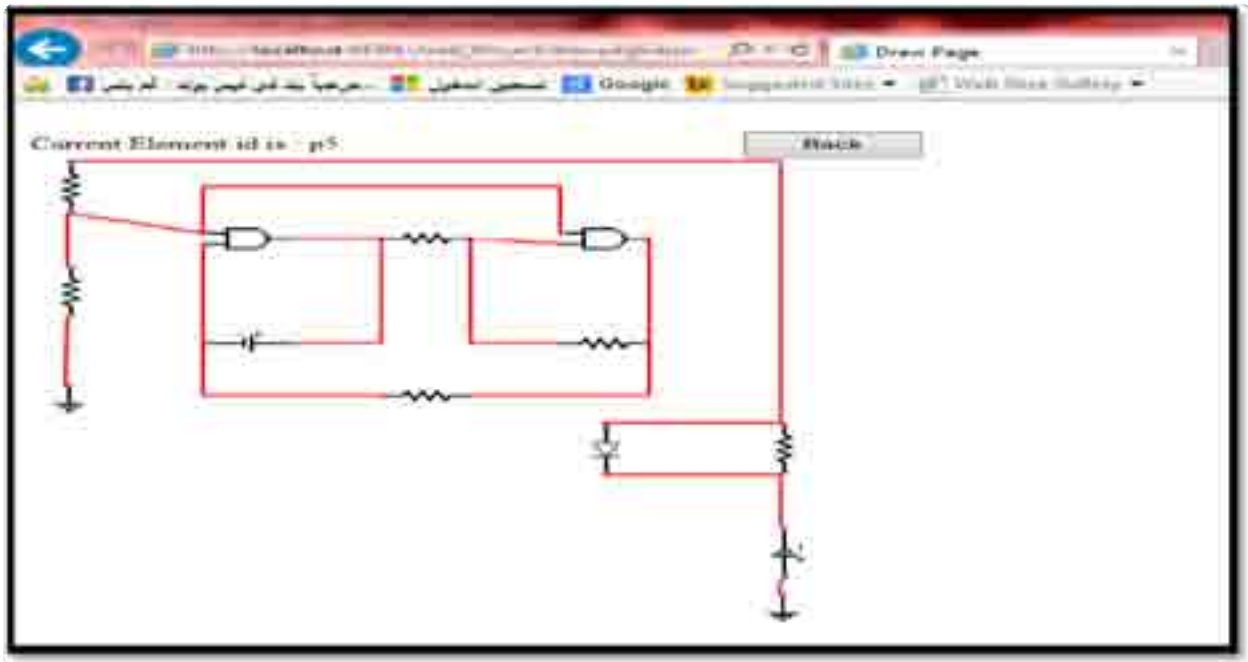
1 <RES id='r1' pos='0' consider='r1:r1,r1:r1' rotate='true' > </RES>
2 <RES id='r2' pos='30' consider='r1:r1,r1:r1' rotate='true' > </RES>
3 <PLANE id='p1' pos='1' consider='r1:r1,r1:r1' > </PLANE>
4 <GRID id='g1' pos='10' consider='r1:r1,r1:r1' > </GRID>
5 <RES id='r3' pos='20' consider='r1:r1,r1:r1' > </RES>
6 <PLANE id='p2' pos='8' consider='r1:r1,r1:r1' rotate='true' > </PLANE>
7 <PLANE id='p3' pos='5' consider='r1:r1,r1:r1,r1:r1' > </PLANE>
8 <GRID id='g2' pos='24' consider='1' > </GRID>
9 <RES id='r4' pos='140' consider='r1:r1,r1:r1' > </RES>
10 <RES id='r5' pos='100' consider='r1:r1,r1:r1' > </RES>
11 <PLANE id='p4' pos='140' consider='r1:r1,r1:r1,r1:r1' > </PLANE>
12 <PLANE id='p5' pos='147' consider='r1:r1,r1:r1,r1:r1' > </PLANE>
13 <PLANE id='p6' pos='183' consider='r1:r1,r1:r1,r1:r1' > </PLANE>
14 <RES id='r6' pos='190' consider='1' > </RES>
15 <PLANE id='p7' pos='120' consider='r1:r1,r1:r1,r1:r1' > </PLANE>
16 <GRID id='g3' pos='190' consider='1' > </GRID>
17 <RES id='r7' pos='244' consider='1' rotate='true' > </RES>
18 <RES id='r8' pos='248' consider='r1:r1,r1:r1,r1:r1,r1:r1' rotate='true' > </RES>
19 <RES id='r9' pos='348' consider='r1:r1,r1:r1' rotate='true' > </RES>

```

Draw Help

Figure (4.10): XML code written by user to draw the circuit.

The Electrical circuit after pressing the draw button.



**Figure (4.11): electrical drawing of our project.**

### 4.3 Limitations:

- 1- Connection with internet.

This model needs internet connection to draw the electrical circuits.

- 2- Easy to use.

With proposed model the user should write descriptive code to get circuit, but with drawing programs the user only drags drop the element of work space. We can solve this limit by doing generating code for element in future work and then we can get the model competitive the software programs.

## Chapter Five

## Conclusion & Future Work

### 5.1 Conclusion

Electrical engineering faces some problems in sharing and exchange information and store information over the web, the proposed model directed to solve this problem, by creating a new markup language that is designed for electrical engineering, the model designed for drawing circuit base descriptive code. The user writes XML code base a new markup language, then presses the button (Draw) the code converts to image (an electrical circuit). This image represents the electrical circuit, but this image is sent as text (XML file) not like (GIF·JPEG·PNG), since it's based on the XML document.

This model adds more flexibility in drawing. Software programs had been compared with proposed model, and we get the significant results on image size and space on disc, the size of the image with software program doubles (2411) rather than proposed model, this result means speed of transferring data. In addition, with proposed model we don't need programs to draw, only we need connection with internet and web browser, the new model solves the platform dependency problem, one of the most important point is drawing with proposed model is done on SVG environment, this point adds more flexibility and accuracy in circuit when scaling it without losing quality. Also with this application the user can modify, add and delete easily without the need to redraw the image because the image.

## **5.2 Future Work**

As a future work, it is highly recommended to do further of development this application. By doing the code generates for elements, if we can do this generated model code, the user can draw by drag and drop such as software programs we can surpass the competitive model with the software programs.

## References

Aktualisiert, Z. (2013 ). *BML-Blackbird Theatrical Services*. **BML in Internet**.

Beezer, J. L., DeMello, M. A., & Silver, D. M. (2013). *U.S. Patent N 8,555,198*. **Washington, DC: U.S. Patent and Trademark Office**.

Berlin, S., Bose, S. B., Inguva, S., & Chhatre, C. M. (2014). *U.S. Patent No. 20,140,047,325*. **Washington, DC: U.S. Patent and Trademark Office**.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1997). *Extensible markup language (XML)*. **World Wide Web Journal, 2 (4), 27**.

*Chemical Markup Language*. de Jong (2013).et al. **Journal of Cheminformatics**

Das, B. (2005). *XML AND ITS APPLICATION IN INFORMATION SYSTEM: AN OVERVIEW*. **Convention PLANNER, 278**.

David Carlisle (NAG), P. I. (2000). *W3C liability, trademark, document use and software licensing rules apply*. , 50.

Drath, R., Luder, A., Peschke, J., & Hundt, L. (2008, September) AutomationML-the glue for seamless automation engineering. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008*. **IEEE International Conference on (pp. 616-623). IEE**



Debreceeny, R. S. -A. (2005). *Financial Reporting in XBRL on the SEC's EDGAR System: A Critique and Evaluation*. **Journal of Information Systems**.

Etherington, G. J., & MacLean, D. (2013). *SVGenes: a library for rendering genomic features in scalable vector graphic format*. **Bioinformatics**, *29* (15), 1890-1892.

Fallside, D. C., & Walmsley, P. (2004). *XML schema part 0: primer* second edition. **W3C recommendation**.

Frank Atanassowa, J. J. (2006). *Customizing an XML–Haskell data binding with type isomorphism*. **Elsevier B.V.** 73.

Fredw. (2011). *Arabic mathematics in Mathzilla*. **Blog de Frédéric**, 2.

Florin, D, Daniel, V, & Florina, P. A. (2013). Financial Reporting under XBRL and the Impact on the Financial Audit. *Ovidius University Annals, Series Economic Sciences*, *13* (1).

Grijzenhout, S., & Marx, M. (2013). *The quality of the XML web*. *Web Semantics: Science, Services and Agents on the World Wide Web*, *19*, 59-68.

Harvey, Richard Hans and Gardiner, Benjamin Michael (2013). *System and method for providing a directory service network*. **Google Patents. US Patent 8,572,201**

Hamscher, W. (20 April 2004). *XBRL Formula Requirements*. **XBRL INTERNATIONAL PUBLIC WORKING DRAFT.**

Huang, Z., Eliëns, A., & Visser, C. (2003, March). *Implementation of a scripting language for VRML/X3D-based embodied agents*. In *Proceedings of the eighth international conference on 3D Web technology* (pp. 91-100). ACM.

Hundt, L., Drath, R., Lüder, A., & Peschke, J. (2008, June). *Seamless Automation Engineering with AutomationML®*. In *Proceedings of the 14th International Conference on Concurrent Enterprising IEC, S* (pp. 685-692).

H. M. Deitel, p. (2003). *XML how to program*. New jersey: **prentice hall.**

Holzner, S. (2003). *Teach Yourself XML in 21 Days (3rd Edition Ed.)*. **SAMs.**

IBM, (2009). *DB2 Universal Database Version 8 is going out of support as of April 30, 2009.*

Jakub Klímeš, M. N. (2012). *On Inheritance in Conceptual Modeling for XML*.  
**Published by Elsevier Ltd.**

Jason R Swedlow, Gianluigi Zanetti & Christoph Best *Channeling the data deluge* © 2011 Nature America, Inc. All rights reserved page nature  
**methods** | VOL.8 NO.6 | JUNE 2011 | **463**

Kumar, M., Kothari, P., & Sahni, S. (2013). *U.S. Patent No. 8,515,908*.  
**Washington, DC: U.S. Patent and Trademark Office.**

Landau, R. H., Vediner, D., Wattanakasiwich, P., & Kyle, K. R. (2002). *Future scientific digital documents with MathML, XML, and SVG*. **Computing in Science & Engineering**, 4 (2), 77-85.

Liu, S. M., Li, X., Liao, Q. H., & Wang, L. (2013). *PLM-Oriented Interactive 3D Virtual Exhibition of Special Vehicle*. **Advanced Materials Research**, 644, 370-373.

Ion, P. a. (1998). *Mathematical markup Language (mathml) 1.0 specification*.  
**World wide web consortium (w3c)**

Lu, W., Chiu, K., & Pan, Y. (2006, September). *A parallel approach to XML parsing*. In *Grid Computing, 7th IEEE/ACM International Conference on* (pp. 223-230). IEEE.

Martin, E. (1999). *A Visual Languages for XML*. **Oregon State University** page3.

Mohan, N. (2010). *What is SVG, and Why it's so popular*. **Lightrains Technolabs**.

Mong, J. C., & Brailsford, D. F. (2003, November). *Using SVG as the rendering model for structured and graphically complex web material*. In **Proceedings of the 2003 ACM symposium on Document engineering** (pp. 88-91). ACM.

Murata, M., Laurent, S. S., & Kohn, D. (2001). *XML media types*. **RFC3023**, January.

Murray-Rust P, R. H. (2003). *Chemical markup, XML, and the World Wide Web*. 4. CML schema. National Center for Biotechnology Information, **U.S. National Library of Medicine**.

Olsson, T. (September 26, 2008). *CSS absolute and fixed positioning*. **Attribution-NonCommercial-ShareAlike 2.5** Generic.

PATRICK CAREY, M. K. (2003). *Creating a web page with html and XML*. **United states of America: Thomson course technology**.

Pal, S., Cseri, I., Seeliger, O., Schaller, G., Giakoumakis, L., & Zolotov, V. (2004, August). *Indexing XML data stored in a relational database*. In

*Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 1146-1157). VLDB Endowment.

Peter Murray-Rust 1, H. S. (2000). *Development of Chemical Markup Language (CML) as a System for Handling Complex Chemical Content*. **School of Pharmaceutical Sciences, University of Nottingham, UK.**

Peter, L. (2003). *The Extensible Markup Language*. **academ.edu.**

Pacifici, G., & Youssef, A. (2001). *Patent and Trademark Office. U.S. Patent No. 6,230,171*. **Washington, DC: U.S.**

Schmelzer, R., & Vandersypen, T. (2002). *XML and Web services unleashed*. **SAMs.**

Syntax, X. S. (2002). *Processing, W3C Recommendation*. **Http: /www. w3. Org/TR/XML enc-core/. 2002-12-10.**

Trappey, A. J., Liu, T. H., & Hwang, C. T. (1997). *Using the EXPRESS data modeling technique for PCB assembly analysis*. **Computers in industry, 34 (1), 111-123**

Sprott, D. (2000). *Enterprise resource planning: componentizing the enterprise application packages*. **Communications of the ACM**.

Stephen Buswell, S. D. (7 July 1999). *Mathematical Markup Language (MathML) 1.01 Specification*. Mathematical Reviews / **American Mathematical Society**).

Tom Myer.2005 *A Really, Really, Really Good Introduction to XML Programming, XSLT 2014* **SitePoint Pty. Ltd**.

Walsh, n. (2000). *A technical introduction to XML*. 4 , 49.

Wibe A de Jong, Andrew M Walkerand Marcus D Hanwell (2013). *From data to analysis: linking NWChem and Avogadro with the syntax and semantics of* Zavlavsky, I.

zepa, P. M. -R. (2001). *Chemical Markup Language*. XML- **cml.org**.

<http://codemirror.net/>

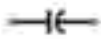
<http://www.w3schools.com>.


<http://www.w3.org/>.







## APPENDICES


### Appendix A:

Classification for electrical symbols.





Symbol	Component name	Meaning
<i>Capacitor Symbols</i>		
	CAP Capacitor	Capacitor is used to store electric charge. It acts as short circuit with AC and open circuit with DC.








Symbol	Component name	Meaning
<i>Inductor / Coil Symbols</i>		
	Inductor	Coil / solenoid that generates magnetic field








<i>I</i>		
Symbol	Component name	Meaning
<i>Power Supply Symbols</i>		
	AC Voltage Source	AC voltage source
	Generator	Electrical voltage is generated by mechanical rotation of the generator
	BAT Battery Cell	Generates constant voltage
	Battery	Generates constant voltage
	Controlled Voltage Source	Generates voltage as a function of voltage or current of other circuit element.
	Controlled Current Source	Generates current as a function of voltage or current of other circuit element.



Symbol	Component name	Meaning
<i>Antenna Symbols</i>		
	Antenna / aerial	Transmits & receives radio waves





Symbol	Component name	Meaning
<i>Meter Symbols</i>		
	V Voltmeter	Measures voltage. Has very high resistance. Connected in parallel.
	AM Ammeter	Measures electric current. Has near zero resistance. Connected serially.
	Ohmmeter	Measures resistance
	Wattmeter	Measures electric power

Symbol	Component name	Meaning
<i>Misc. Symbols</i>		
	<b>M</b> Motor	Electric motor
	Transformer	Change AC voltage from high to low or low to high.
	Electric bell	Rings when activated
	Buzzer	Produce buzzing sound
	Fuse	The fuse disconnects when current above threshold. Used to protect circuit from high currents.
	Bus	Contains several wires. Usually for data / address.
	Optocoupler / Opto-isolator	Optocoupler isolates connection to other board

	Optocoupler / Opto-isolator	Optocoupler isolates connection to other board
	Loudspeaker	Converts electrical signal to sound waves
	Microphone	Converts sound waves to electrical signal
	Operational Amplifier	Amplify input signal
	Analog-to-digital converter (ADC)	Converts analog signal to digital numbers
	Digital-to-Analog converter (DAC)	Converts digital numbers to analog signal
	Crystal Oscillator	Used to generate precise frequency clock signal

Symbol	Component name	Meaning
<b>Lamp / Light Bulb Symbols</b>		
	Lamp / light bulb	Generates light when current flows through
<b>Diode / LED Symbols</b>		
	Diode	Diode allows current flow in one direction only (left to right).

Symbols	Component name	meaning
<b>Transistor Symbols</b>		
	NPN Bipolar Transistor	Allows current flow when high potential at base (middle)
	PNP Bipolar Transistor	Allows current flow when low potential at base (middle)