MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

SOFTWARE REQUIREMENTS SPECIFICATIONS

Date of Issue:

13.01.2016

Project Advisor:

Dr. Onur Tolga ŞEHİTOĞLU

Team Name:

ARCADIA

Team Members:

Mehmet AKALIN - 1880871

Ertuğ UFUK - 1881572

Elif ŞAHİN - 1881846

Merve Gizem ŞENEL - 1881879

Contents

1. Introduction

This document is a software requirement specification for a multi purpose flowchart interpreter. In this document, first we define the problem and give the system overview. Secondly, we give an overall description. Finally, we state specific requirements and data models.

1.1. Problem Definition

This project aims to overcome the natural obscurity of a conventional source code. Understanding and optimizing a program's functionality can be confusing and time-consuming solely by looking at a wall of text. Our purpose is to produce a web-based tool for visualizing a program written with various languages to give users a bird's-eye view of their own product and allow them to trace and modify it with a graphical interface and even build something new from scratch with said GUI.

1.2. System Overview

The product of our project is a visual programming IDE which has a text/source-code editor, build automation tools, visual debugger and graphical representations of data-structures.

Users write their codes with the editor and after validation, it is compiled and converted to a flowchart showing the structure and the process flow of that code. Given visual chart gives core information about the program and also user interface allows modification of that chart, consequently creating a feature to change the initial code with visual control. Moreover, this visual interface can be used to build a flowchart so it is possible to create an enitre program without writing a single line of code.

In addition, debugging and apparent data structures are also visualized so the flow of the program can be interactively traced. Thus, the overall system becomes quite comprehensive to work with.

1.3. Definitions, Acronyms, and Abbreviations

GUI	Graphical User Interface
IDE	Integrated Development Environment
SRS	Software Requirement Specification
Flow Chart	A formalized graphic representation of a logic sequence, work or manufacturing process, organization chart or similar formalized structure.

1.4. Assumptions and Dependencies

- We assume that Clang is sufficent to parse through source codes of the programming languages C, C++ and Objective-C.
- The GNU Project Debugger aka. GDB is assumed to be used for debugging for most, if not every language we work with.
- For our every need in visualizing on a web browser, we will not need any other Javascript library other than JointJS with its plugins.
- Our product will work properly on every up-to-date browser.

2. Overall Description

2.1. Product Functions

2.1.1. Use-Case Model Survey

Name	Actor	Description
Write Code	User	writes their code with the web-site's editor.
Load From File	User	loads their own existing source code file to the text editor
Submit Code	User	submits the code in the text editor to be sent to server
Add Object	User	adds an object from palette to flowchart canvas by drag and drop
Move Object	User	drags the object on the canvas and attached links follow
Delete Object	User	deletes an object from the flowchart canvas
Add Link	User	adds a link between two objects in the canvas
Move Link	User	changes the object to which a link's head is attached on the canvas
Delete Link	User	deletes a link on the canvas
Modify Object	User	adds or clicks on an object, then a form pops up to fill/change information about it
Submit Flowchart	User	submits the flowchart in the canvas to be sent to server
Run Debugger	User	runs the debugger with predefined options
Set Breakpoint	User	sets a breakpoint at the program for debugging
Show Data Structure	User	requests the visual representation of implemented data structures
Manage System	Admin	manages overall system,database, connections etc.

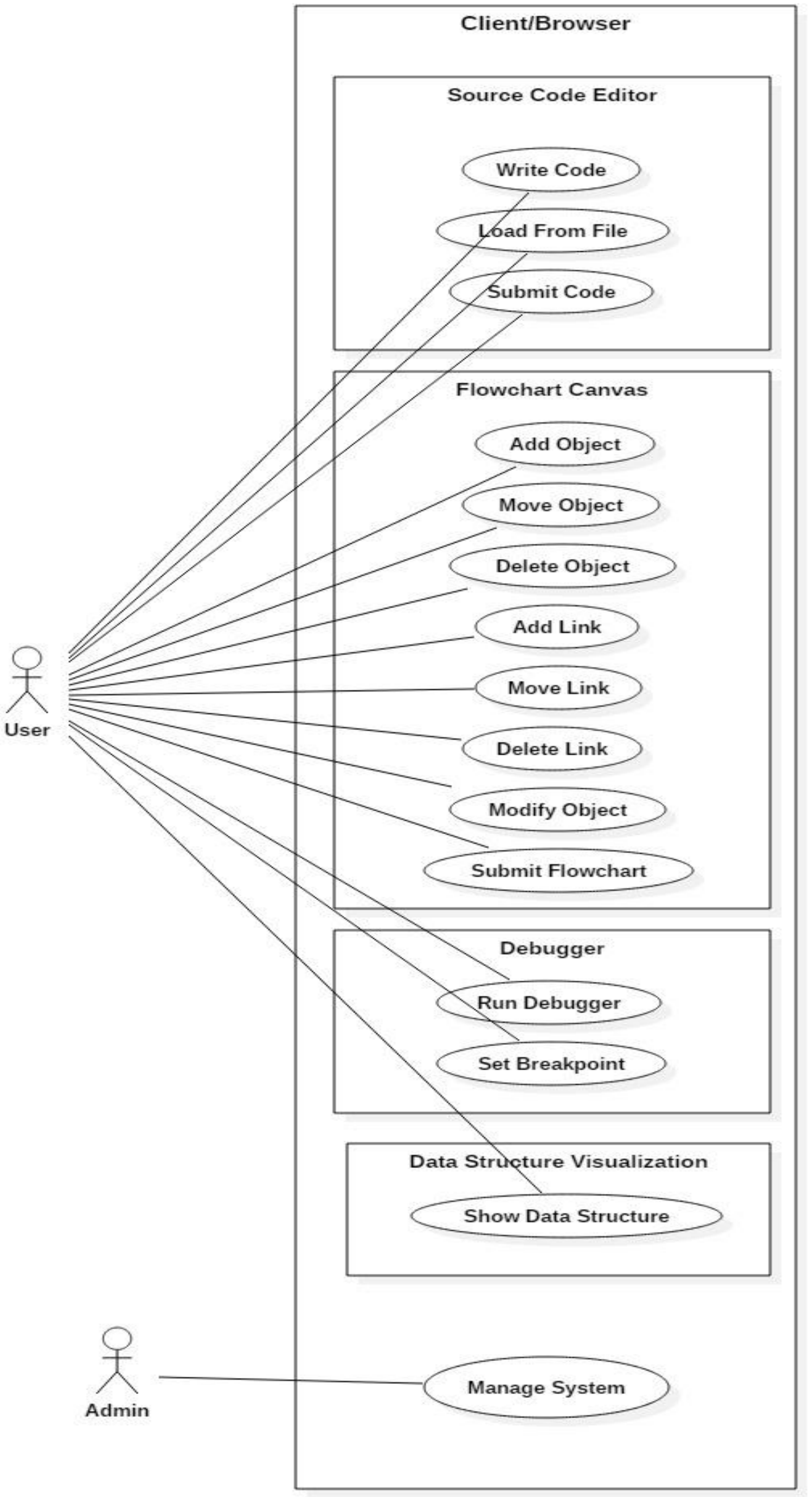


Figure – Use Case Diagram

2.1.2. Actor Survey

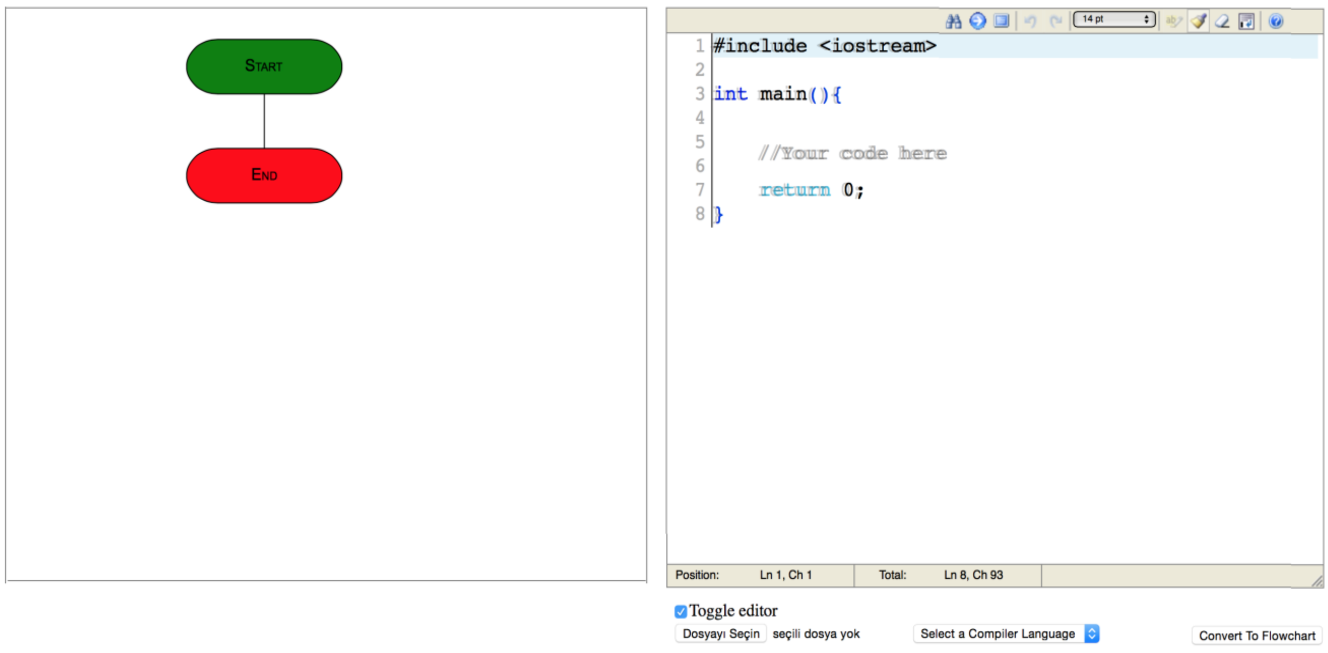
User: Defines people who visit the web-site of FlowCode to submit some code or flowchart and possibly debug their program. Such users are assumed to have basic knowledge about algorithms and programming.

Admin: Overseeing the performance of the system, admin manages the software by installing updates if necessary.

2.2. Interfaces

2.2.1. User Interfaces

Our project is a web-based IDE so user interface is a single web page which is split into two sections, flowchart canvas and source code editor. Users can load, build and modify flowcharts on the canvas and code on the editor. It will also have interfaces to register,login and view user profile. Any browser that is up to date can be used to open this site. Below are some images showing what our web page will look like:



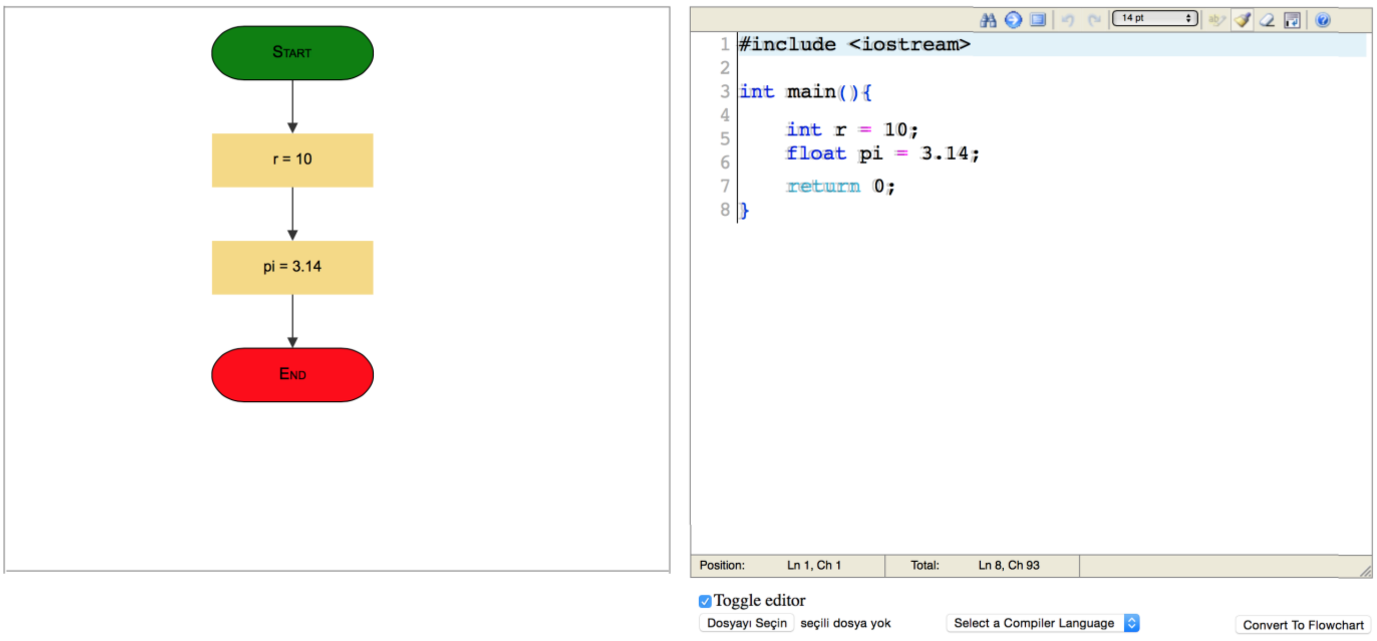
```
#include <iostream>

int main(){

    //Your code here
    return 0;
}
```

Position: Ln 1, Ch 1 Total: Ln 8, Ch 93

Figure – Initial Screen of Our Web Page



```
#include <iostream>

int main(){

    int r = 10;
    float pi = 3.14;
    return 0;
}
```

Figure – Basic assignmet statements and its corresponding flow chart

2.2.2. Hardware Interfaces

The system is purely software, which means it does not possess any hardware interfaces.

2.2.3. Software Interfaces

Submitted codes and flowcharts are sent from client to server in order to be processed, converted and sent back. With the purpose of making the client as lightweight as possible, parsing, compiling and debugging are fully handled by the server and pushed to client. Registered users can also upload and access codes via system database.

2.2.4. Communication Interfaces

Any communication between server and client is achieved via JSON.

2.3. Constraints

- Front-end design constraints consist of using HTML5, Javascript with JointJS library and its plugins and JQuery/Ajax.
- Back-end constraints consist of using Python and Django.

3. Specific Requirements

3.1. Functional Requirements

Name:	Write Code
Description:	User should be able to write and edit their code on the text editor.
Data Flow:	User input is shown on the client/browser.
Requirement:	No pre-condition.

Name:	Load Code from File
Description:	User can load existing source code file to the editor.
Data Flow:	File in user repository is sent to client.
Requirement:	User must have an already existing code file.

Name:	Select Compiler Language
Description:	User should be given the choice of selecting compiler language.
Data Flow:	Compiler language information to be sent to server is assigned by the user.
Requirement:	No pre-condition.

Name:	Submit Code
Description:	User may initiate submission of the code any time they want.
Data Flow:	Anything on the editor is sent to server as a JSON object.
Requirement:	No pre-condition.

Name:	Validate Code
Description:	User should see if their submission caused a compile-time error for the selected language.
Data Flow:	Text in the editor is sent to the server to be checked for any compile errors.
Requirement:	-User must make a submission. -User must have selected the compiler language.

Name:	Show Flowchart
Description:	User will see their code as a generated flowchart on the canvas
Data Flow:	The source code that was sent to the server is parsed and converted into a JSON object which is then sent to the client to be visualized.
Requirement:	-Submitted code must have passed validation.

Name:	Build Flowchart
Description:	User can build a flowchart by starting from scratch or by modifying an existing one on the canvas with the object palette of the user interface.
Data Flow:	User input is shown on the browser/client.
Requirement:	To modify a flowchart, user first needs to load one.

Name:	Submit Flowchart
Description:	User can submit the flowchart on the canvas to get its source code equivalent of the desired programming language on the text editor.
Data Flow:	The flowchart as a Javascript object is sent to server to be translated and sent back to client as a source code.
Requirement:	No pre-condition.

Name:	Debug
Description:	User can turn on debugging mode to set breakpoints, run step by step etc. and see the visual flow of their program by the animated flowchart.
Data Flow:	Debugging is done on the server. Hence, each iteration of the debugging process is done with information trade between client and server.
Requirement:	There needs to be an existing, validated code to debug it.

Name:	Visualize Data Structures
Description:	User should be able to see the data structures which they implemented in their code as visual blocks.
Data Flow:	The data structures are processed via pointers in the server.
Requirement:	There must be data structures implemented in the validated code.

3.2. Nonfunctional Requirements

3.2.1. Usability

The main usability requirement for users to have core knowledge about flowchart structure, whether they want to convert flowchart to code or code to flowchart.

Users who convert their code to flowchart are expected to be somewhat familiar with the programming language they are using. However, an alternate scenario might negate that requirement, where the user uploads a third-party source code.

3.2.2. Reliability

Since FlowCode is based on a web client-server architecture, availability is directly related to server condition and user internet connection. Server overload, maintenance and attacks taken into consideration, the system is assumed to be available 90% of the time and serve for days uninterrupted. Also, degraded mode operations consist of making editions on text editor or flowchart canvas.

This system does not yield critical real-time operations so being out of operation is tolerable for a few days in average.

3.2.3. Performance

Response time between client and server is 10 seconds in average and a minute maximum.

For converting between flowcharts and source codes, the file is transacted as a whole but in debugging and data structure visualization, transactions are handled asynchronously.

When connection between client and server fails, user still can edit code and flowchart.

3.2.4. Supportability

System is compatible for user account feature along with personal and public repository for code and flowchart database to be integrated.

4. Data Model and Description

4.1. Data Description

4.1.1. Data Objects

Class Diagram is shown below:

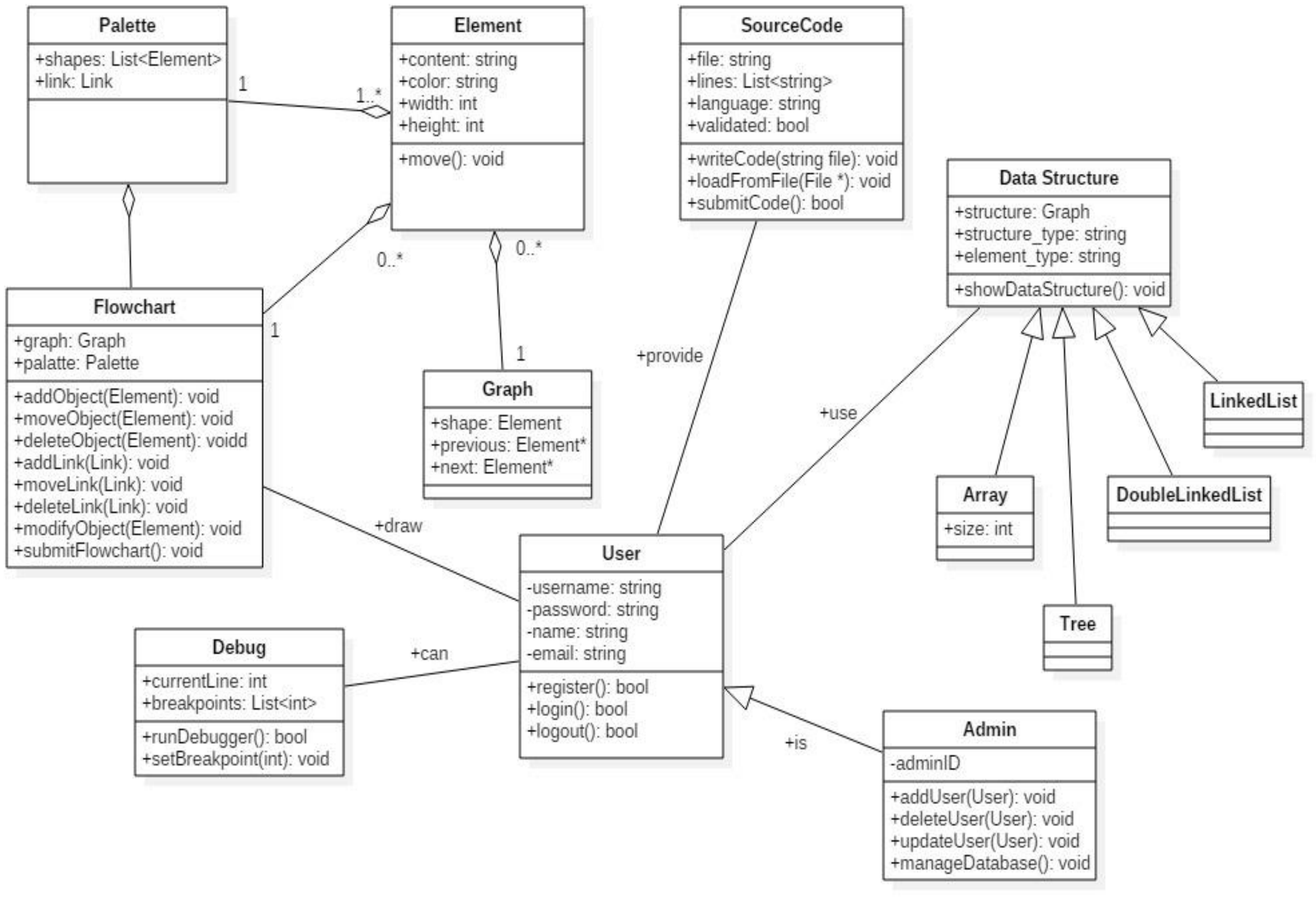


Figure – Class Diagram

4.1.2. Data Dictionary

Data dictionary of the overall system :

Class Name	Description
User	**Users will have username and passwords when they register to the system. They can also provide their e-mail.**
Admin	**Admin is mainly in charge of adding, updating and deleting users. Therefore, his task is to manage the database.**
SourceCode	**Source code provided by the user is stored in a SourceCode object to enable sending this code to the server and to achieve the visual debugging. Lines are used to check whether a breakpoint has been set.**
FlowChart	**FlowChart class holds the flow chart either drawn by the user or the created one from the source code. It has graph and palette attributes for this purpose.**
Debug	**Debug class keeps the current line number of the debugging session and also the breakpoints that are set by the user.**
Data Structure	**Data Structures are processed as graphs and they also keep variables that show what kind of structure they are and what type of element they hold.**
LinkedList	**Link Lists don't have links to their previous elements so we omit the member "previous" while constructing a link list.**
Double Linked List	**Standard graph structure will suffice in the making of a doubly linked list.**
Array	**In addition to our natural graph structure, the array structure will keep the information of its length.**
Tree	**In a tree, an element will only know about its children so we omit the member "previous" while constructing a link list.**
Graph	**Graph embodies the flowchart that is built or converted from code. It holds all the nodes of the flowchart and their relations.**

Palette	**Shapes and link instances reside in palette and they are dragged and initialized from here.**
Element	**Elements are statement objects which populate the flowchart and palette. They hold information about how they are visually shown.**

5. References

[1] IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883