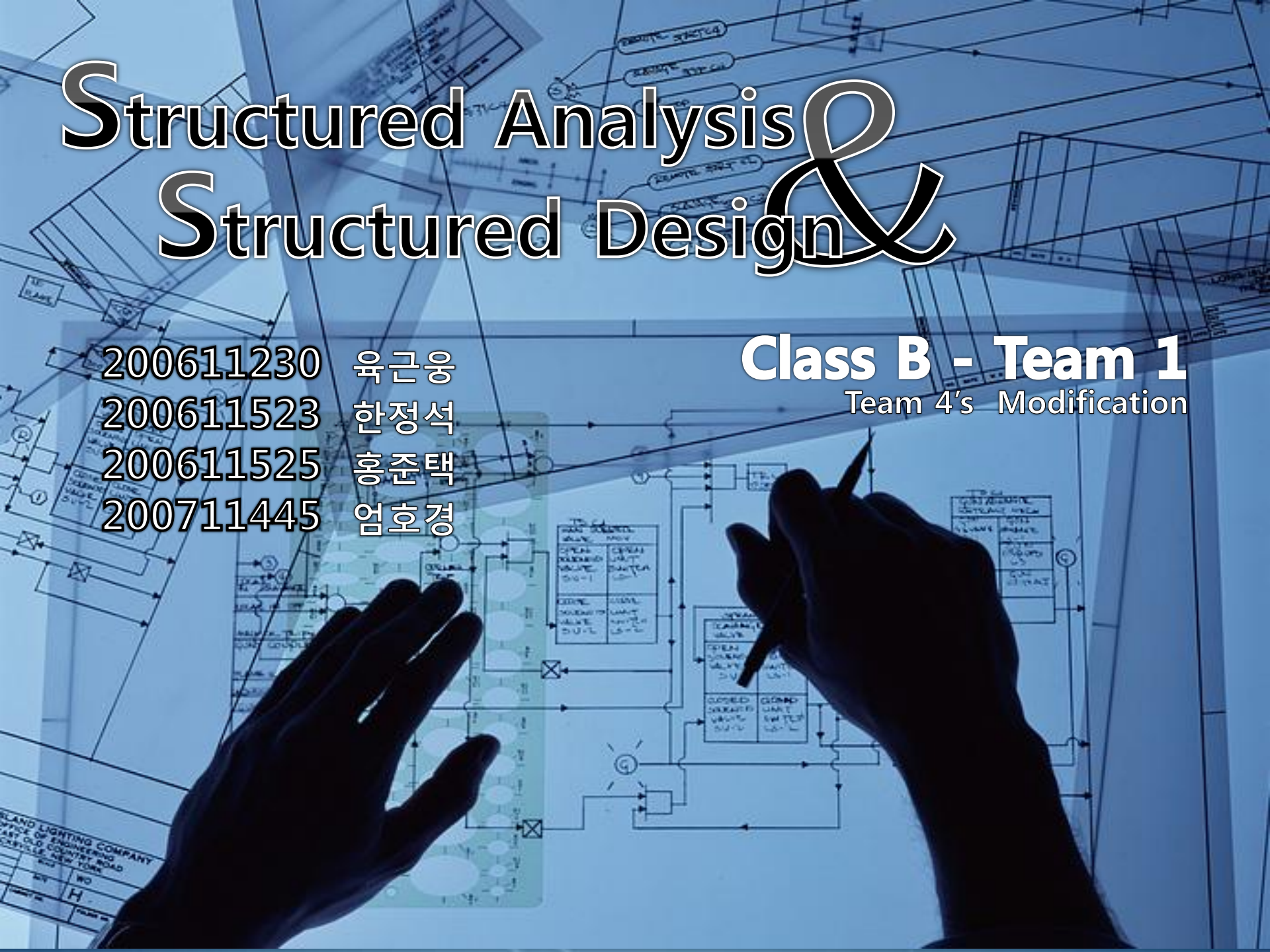


Structured Analysis & Structured Design

200611230 육근웅
200611523 한정석
200611525 홍준택
200711445 엄호경

Class B - Team 1

Team 4's Modification

References

Modern Structured Analysis

Edward Yourdon - 1989

Introduction to System Analysis and Design : a Structured Approach,

Penny A. Kendall - 1996

Structured Analysis and Structured Design (SASD) - Class Presentaion

http://pages.cpsc.ucalgary.ca/~jadalow/seng613/Group/

Zhou Qun, Kendra Hamilton, and Ibrahim Jadalowen - 2002

Contents

Structured Analysis

- Statement of Purpose
- System Context file Diagram
- Event List
- DFD(Data Flow Diagram)
- Data Dictionary
- Process Specification

Structured Design

- Structured Charts –Transform Analysis
- Structured Charts(Basic)
- Structured Charts(Advanced)

Implementation

Demonstration

Statement of Purpose

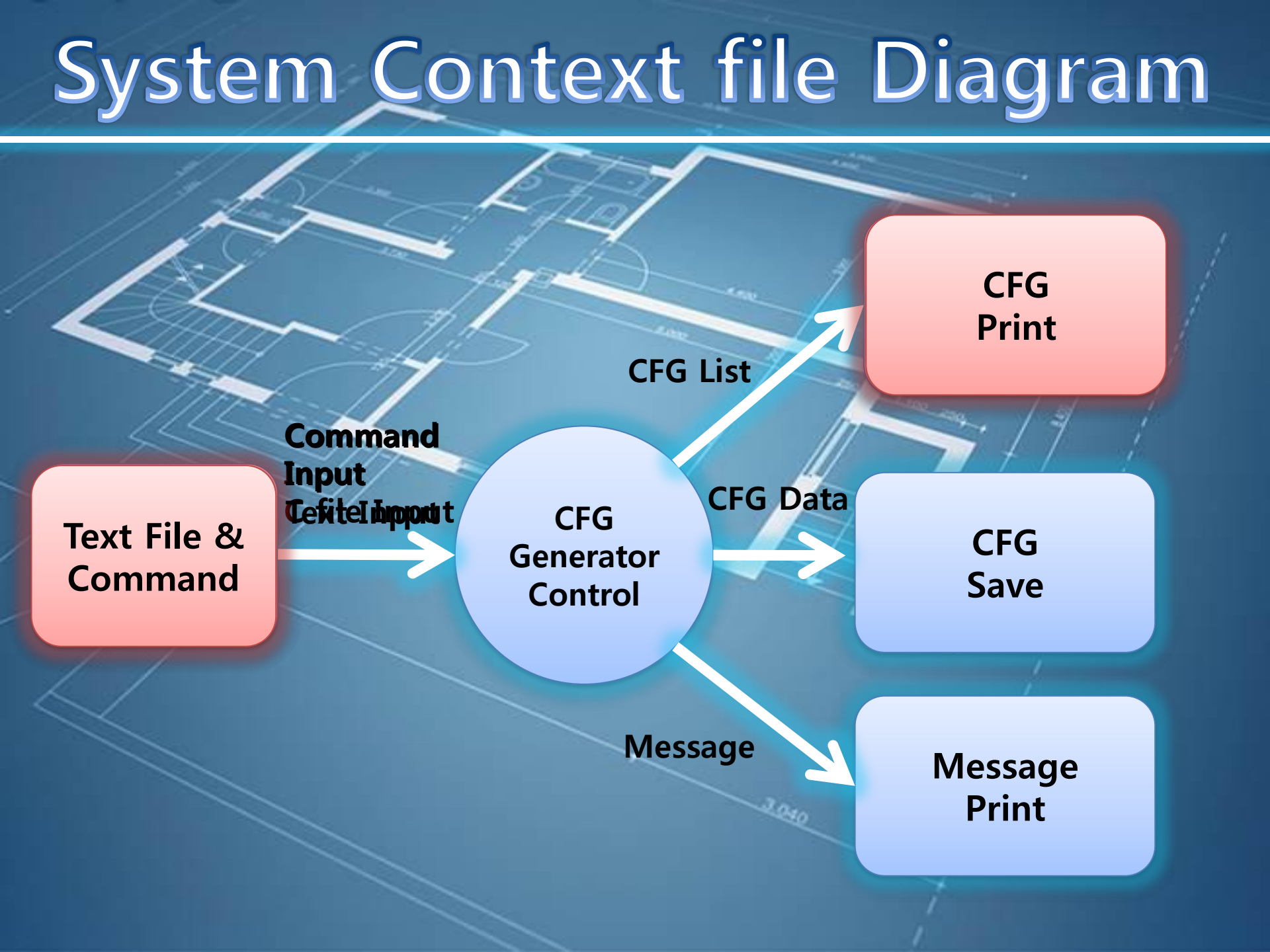
- A program that gets c file and displays CFG
- Input file is restricted to c file that was written in c language and if error occurs, help message will appear
- If user inputs wrong command, help message will appear
- User Interface is featured in CUI format and Command Line Input style
- Command Input(./gc inputcode.c restult.txt) receives 2 argv[] variables and deals rest of inputs as error
- Only mono-filed code will run the program
- Comments will be disregarded
- Let user know the beginning of the transformation before it actually starts
- Control Flow Graph will be displayed in (Node -> Node) format
  and -> stands for expression statement, => for selecting statement, and >> for back edge

Statement of Purpose

Restriction

- Code must be between 100 ~ 200 lines
- Code must have main function
- Code with user-defined header will not accepted
- Code will not have pointers
- Implementation will be executed in Cygwin environment and use gcc compiler

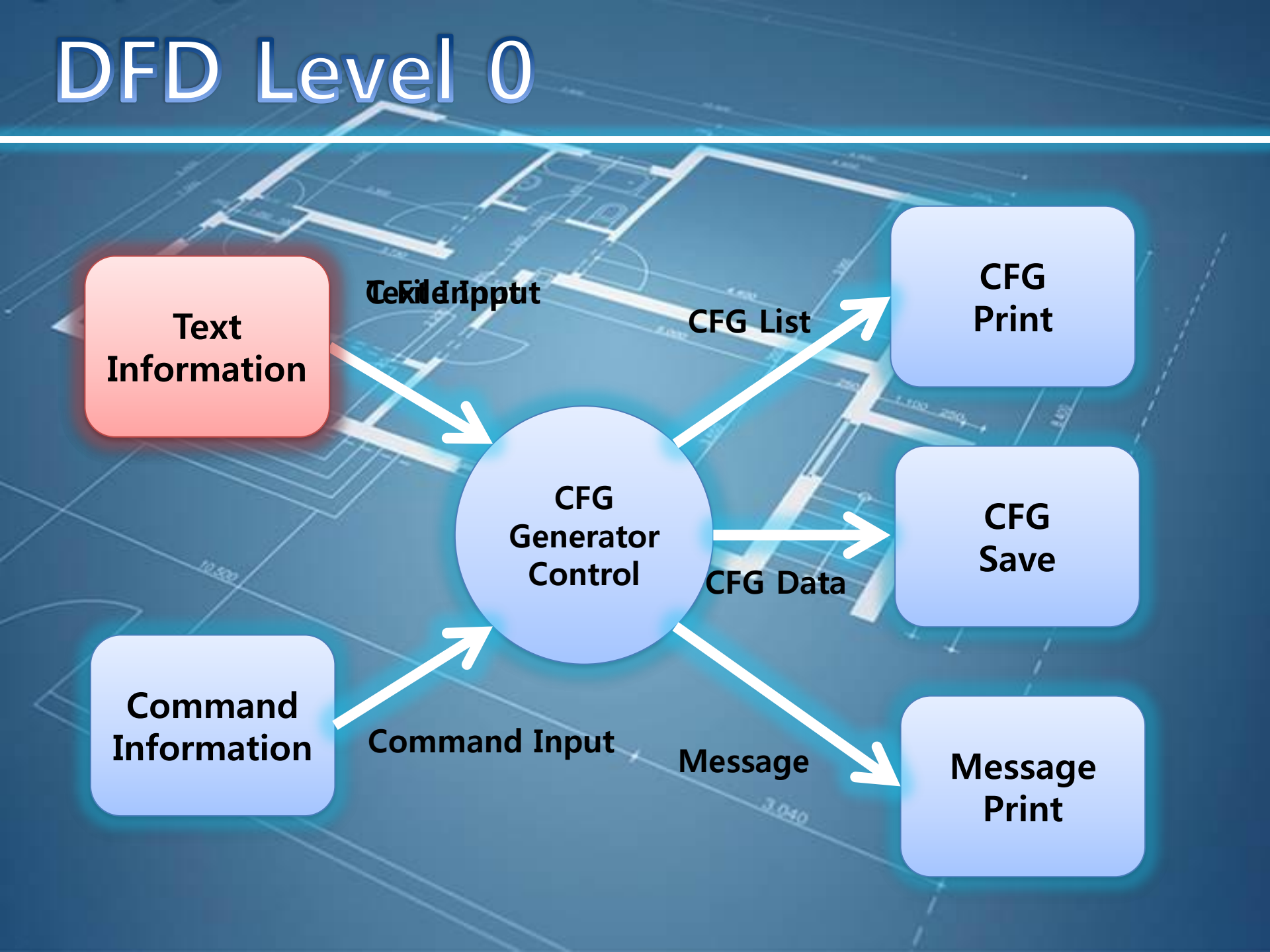
System Context file Diagram



Event List

Input / Output Event	Description
Text Input	Check the Text contents
CFG List Output	Print commands to the CFG Print
Command Input	Check the command
Message Output	Print commands to the Message Print
CFG Data	Save CFG data from CFG generator in TXT format

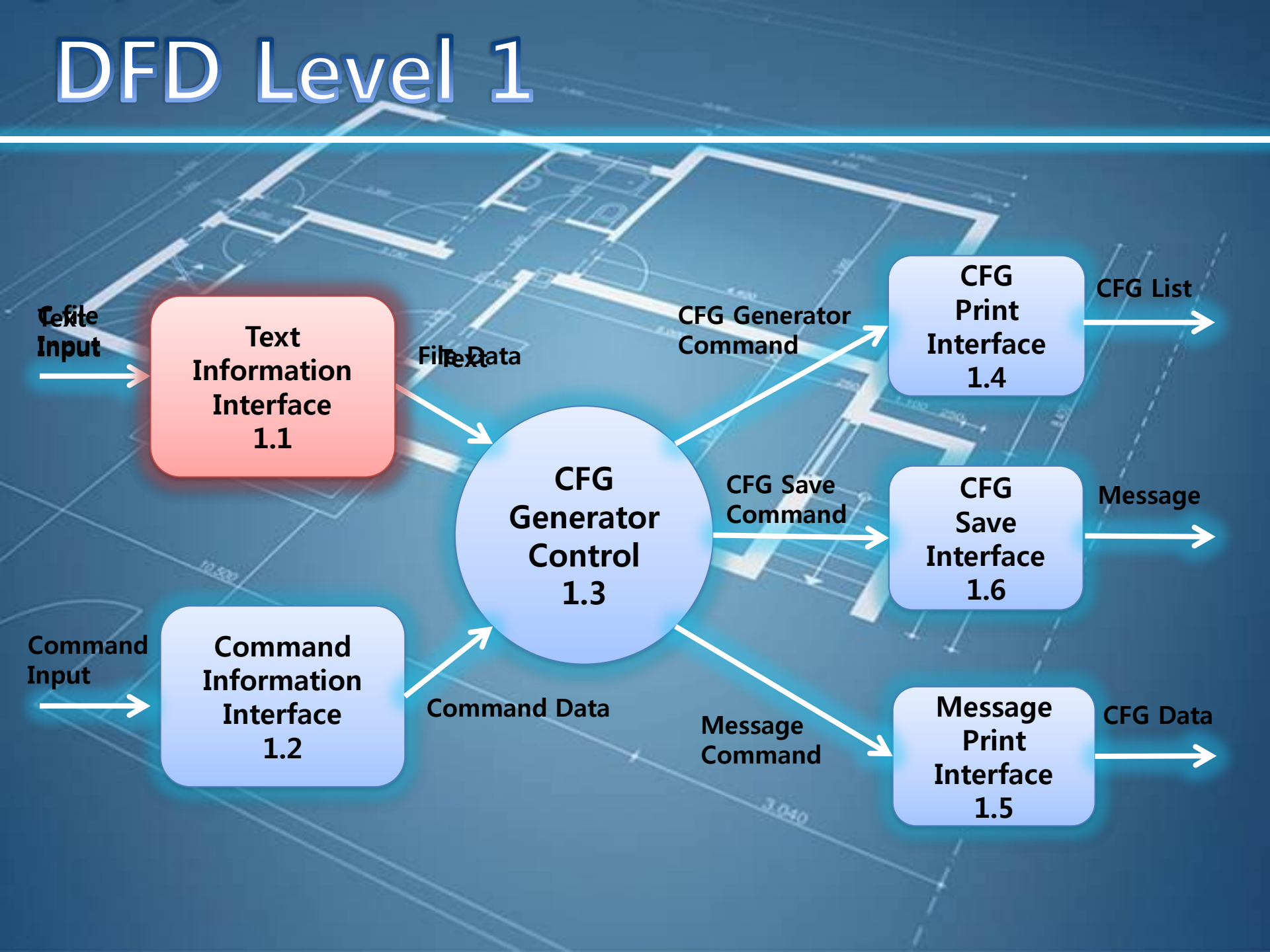
DFD Level 0


Text Information
Text Input
CFG List
CFG Print
CFG Generator Control
CFG Save
CFG Data
Command Information
Command Input
Message
Message Print

Data Dictionary – Level 0

Input / Output	Description	Type
Text Input	CFG to be converted into C code name .txt filed with the extension	.txt
CFG List Output	Print commands to the CFG Print	Char*
Command Input	Check the command	Char*
Message Output	Print commands to the Message Print	Char*
CFG Data	Save CFG data from CFG generator in TXT format.	Txt file

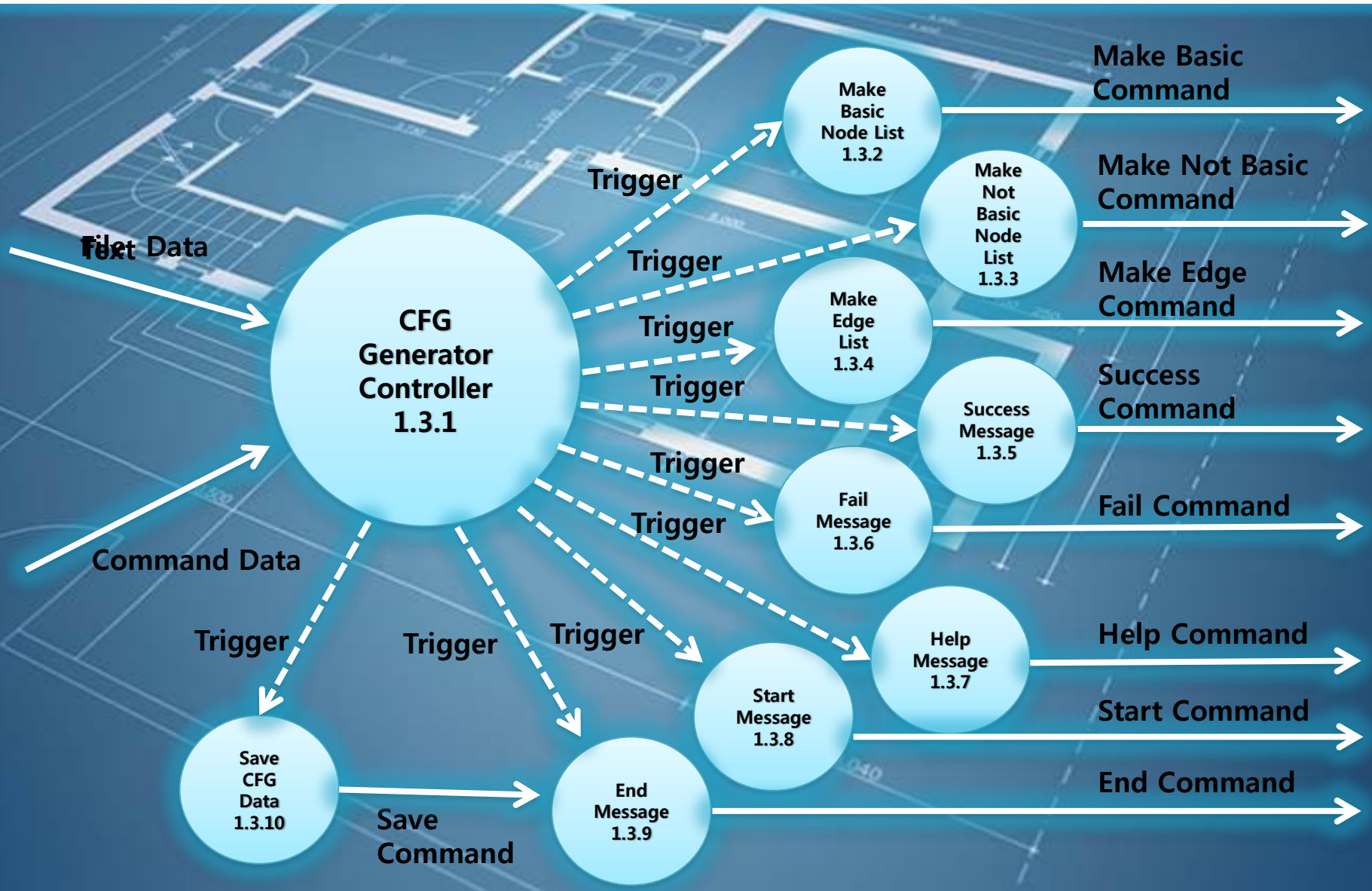
DFD Level 1


Text File Input
Text Information Interface 1.1
File Data
CFG Generator Control 1.3
CFG Generator Command
CFG Print Interface 1.4
CFG List
CFG Save Command
CFG Save Interface 1.6
Message
Command Input
Command Information Interface 1.2
Command Data
Message Command
Message Print Interface 1.5
CFG Data

Data Dictionary – Level 1

Input / Output	Description	Type
Text	Check the Text contents	Char*
Command	A string containing the command	Char*
CFG Command	Distinct node and edge information Is passed	Char[] Char*[]
Message Command	Distinct message to convey Information	A(start Message)/!A(End Message)/C(Success Message)/!C(Fail Message)/Cmd/!Cmd (help Message)
Command	status	Char []
CFG Save Command	Data structure which contains CFG data to be stored in TXT format	Data Structure

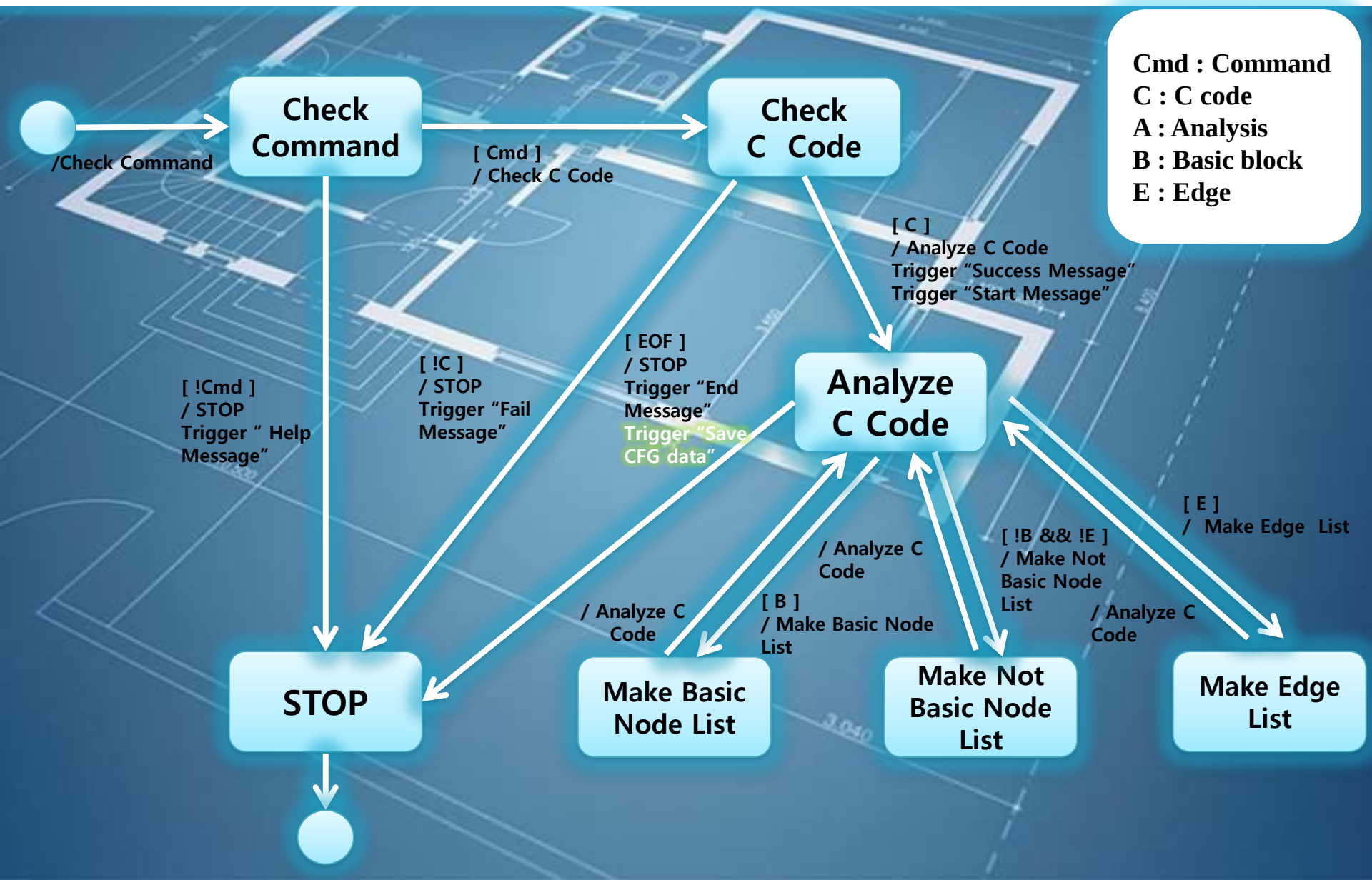
DFD Level 2


Text Data
CFG Generator Controller 1.3.1
Command Data
Trigger
Trigger
Trigger
Trigger
Trigger
Trigger
Trigger
Trigger
Trigger
Make Basic Node List 1.3.2
Make Not Basic Node List 1.3.3
Make Edge List 1.3.4
Success Message 1.3.5
Fail Message 1.3.6
Help Message 1.3.7
Start Message 1.3.8
End Message 1.3.9
Save CFG Data 1.3.10
Make Basic Command
Make Not Basic Command
Make Edge Command
Success Command
Fail Command
Help Command
Start Command
End Command
Save Command

Data Dictionary – Level 2

Input / Output	Description	Type
Make Basic Node Command	Basic node is stored in an array	Char[]
Make Not Basic Node Command	Not Basic node is stored in an array	Char[]
Make Edge Command	Edge is stored in an array	Char*[]
Success Command	The Success of the command is passed	Char*
Fail Command	Enter the file fails to pass	Char*
Help Command	Deliver help message	Char*
Start Command	Analysis of the program began and it is passed	Char*
End Command	Analysis of the program is over and it is passed	Char*
Save Command	A set of data that contains all the information about node and edge lists.	Data Structure

DFD – Level 3


Cmd : Command
C : C code
A : Analysis
B : Basic block
E : Edge
/Check Command
Check Command
[Cmd]
/ Check C Code
Check C Code
[C]
/ Analyze C Code
Trigger "Success Message"
Trigger "Start Message"
[!Cmd]
/ STOP
Trigger " Help Message"
[!C]
/ STOP
Trigger "Fail Message"
[EOF]
/ STOP
Trigger "End Message"
Trigger "Save CFG data"
Analyze C Code
[E]
/ Make Edge List
[!B && !E]
/ Make Not Basic Node List
/ Analyze C Code
/ Analyze C Code
[B]
/ Make Basic Node List
/ Analyze C Code
STOP
Make Basic Node List
Make Not Basic Node List
Make Edge List

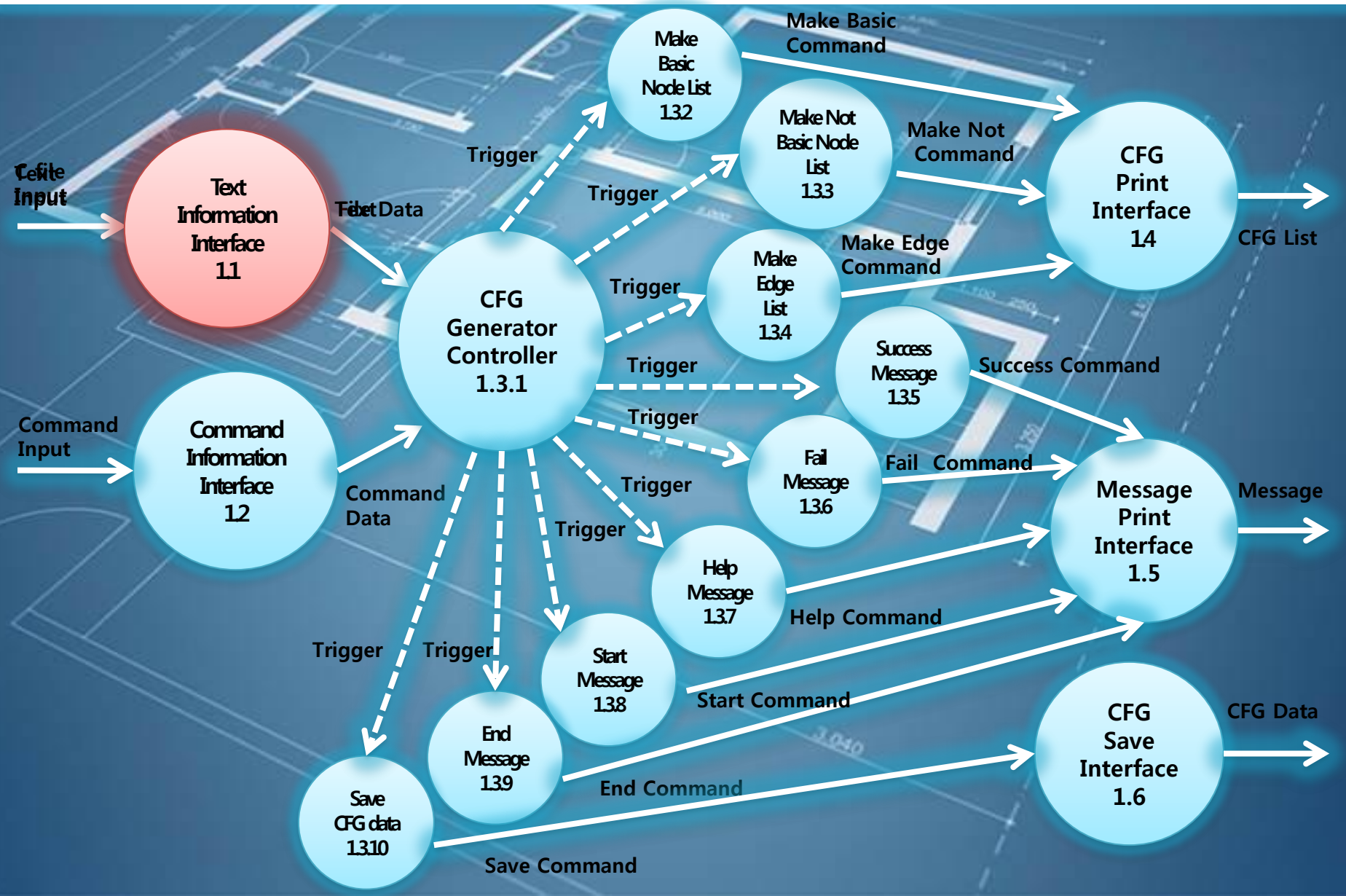
Process Specification

Process	Description
Check Command	Check contents which were inserted from command line whether there are inappropriate keys or not. First argument column must be end with .c format and the second one with .txt format. If either of argument column has a problem, the system will be terminated with a "Help" message.
Check C Command	Check contents of the c file in order to get existence of the file, Line of Code, user-defined header file, presence of its main function, usage of pointers. If the system cannot find the file, or the file has less than 100 lines or more than 200 lines, or the file has user-defined header files or doesn't have main function, or the file uses pointers in its code, the system will be terminated with a "Fail" message. In contrast, if there's nothing wrong with the file, the system will show "Success" message and let users know the beginning of analyzation.
Analyze C Code	Examine each line of the code to determine its statement syntax. If the statement is expression statement, this process will call "Make Basic Node List" function in order to add a node into a basic-node list. If the statement analyzed as an iterating or selecting statement, the process will call "Make Not Basic Node List" function to add a node into a not-basic-node list. When the moment to make edge for nodes has come, the process will call "Make Edge List" function to insert relationship of the node into a edge list. If it's the End of File, the process displays "show" message and calls "Save CFG data".

Process Specification

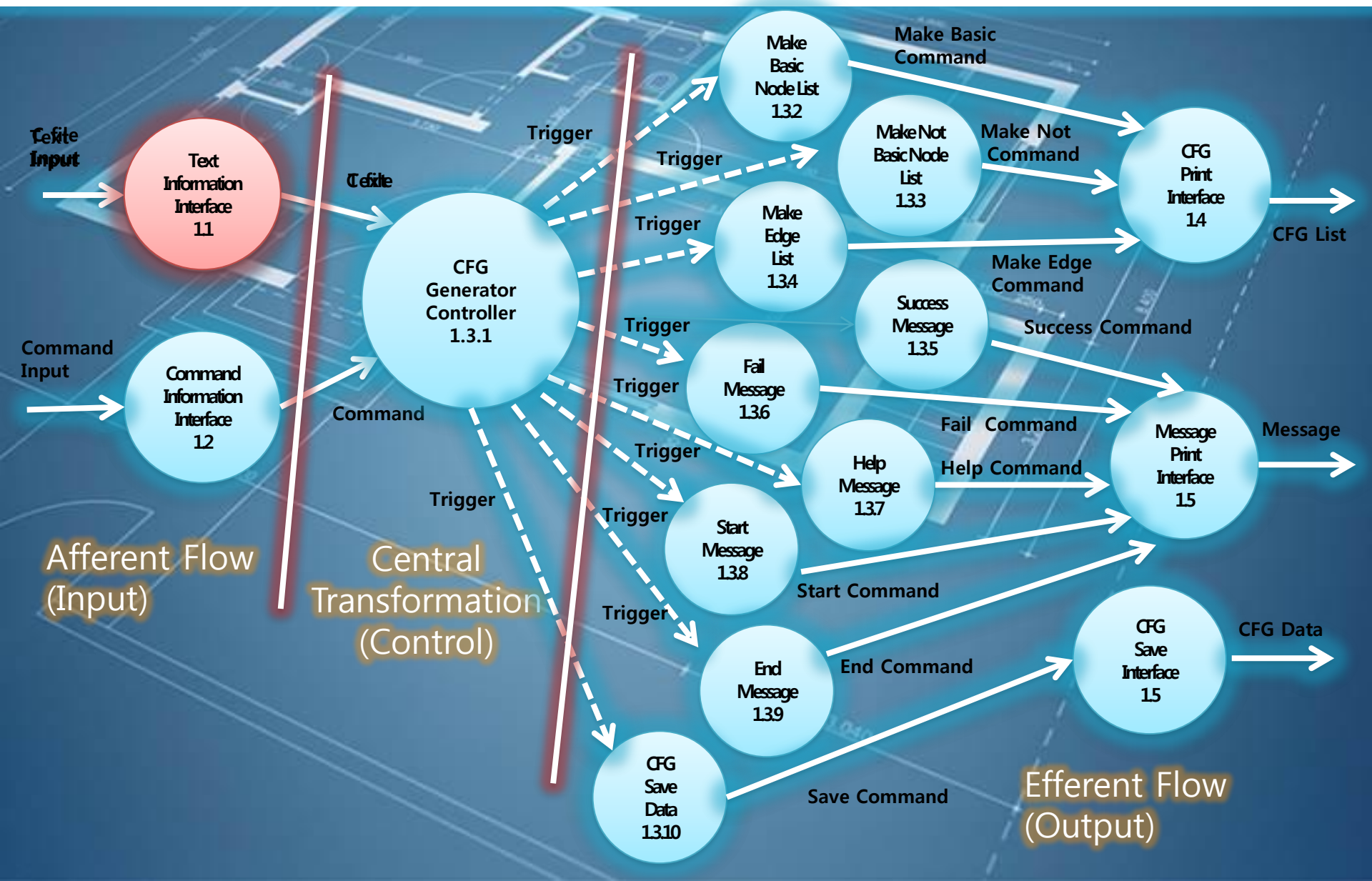
Process	Description
Make Basic Node List	Add a column to a basic-node list. Each column consists of contents of the expression statement and a number of node that increases by 1. the number of node starts from 0. After the action is over, the process calls "CFG Print Interface" to display statements on the screen.
Make Not Basic Node List	Add a column to a not-basic-node list. Each column consists of contents of the iteration or selection statement and a number of node that increases by 1. the number of node starts from 0. After the action is over, the process calls "CFG Print Interface" to display statements on the screen.
Make Edge List	Determine the relation of two nodes that are next to each other in the code and add a column to a edge list. The column contains locations of those nodes and their relationship. After the action is over, the process calls "CFG Print Interface" to display edges on the screen in certain forms following a specific rule.
Stop	Abstract Process that only shows the end of the main controller

DFD – Overall


Text Information Interface 1.1
Command Information Interface 1.2
CFG Generator Controller 1.3.1
Make Basic Node List 1.3.2
Make Not Basic Node List 1.3.3
Make Edge List 1.3.4
Success Message 1.3.5
Fail Message 1.3.6
Help Message 1.3.7
Start Message 1.3.8
End Message 1.3.9
Save CFG data 1.3.10
Message Print Interface 1.5
CFG Save Interface 1.6
CFG Print Interface 1.4
Trigger
Command Input
Command Data
Make Basic Command
Make Not Command
Make Edge Command
Success Command
Fail Command
Help Command
Start Command
End Command
Message
CFG List
CFG Data
Save Command

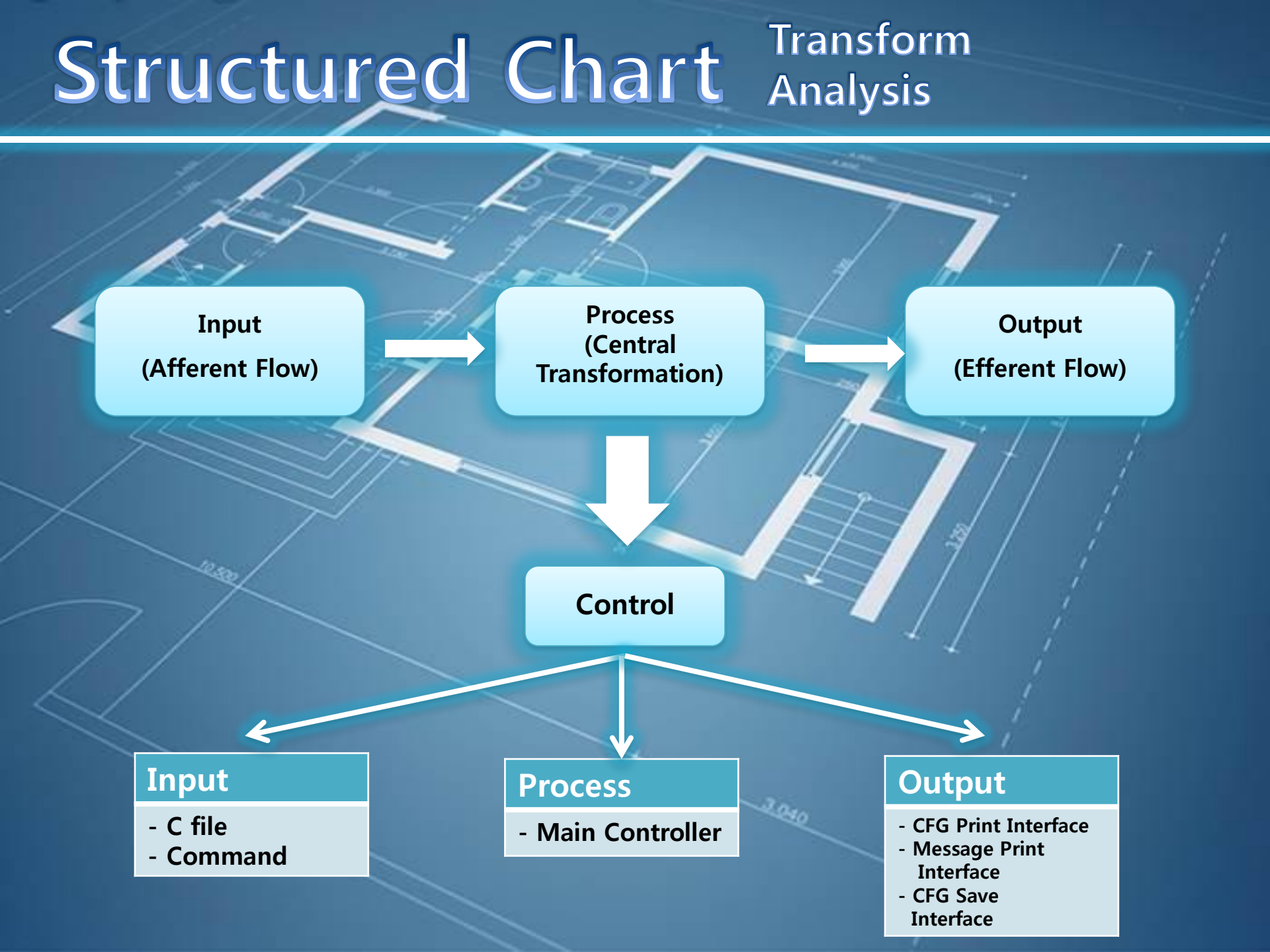
Structured Chart

Transform Analysis



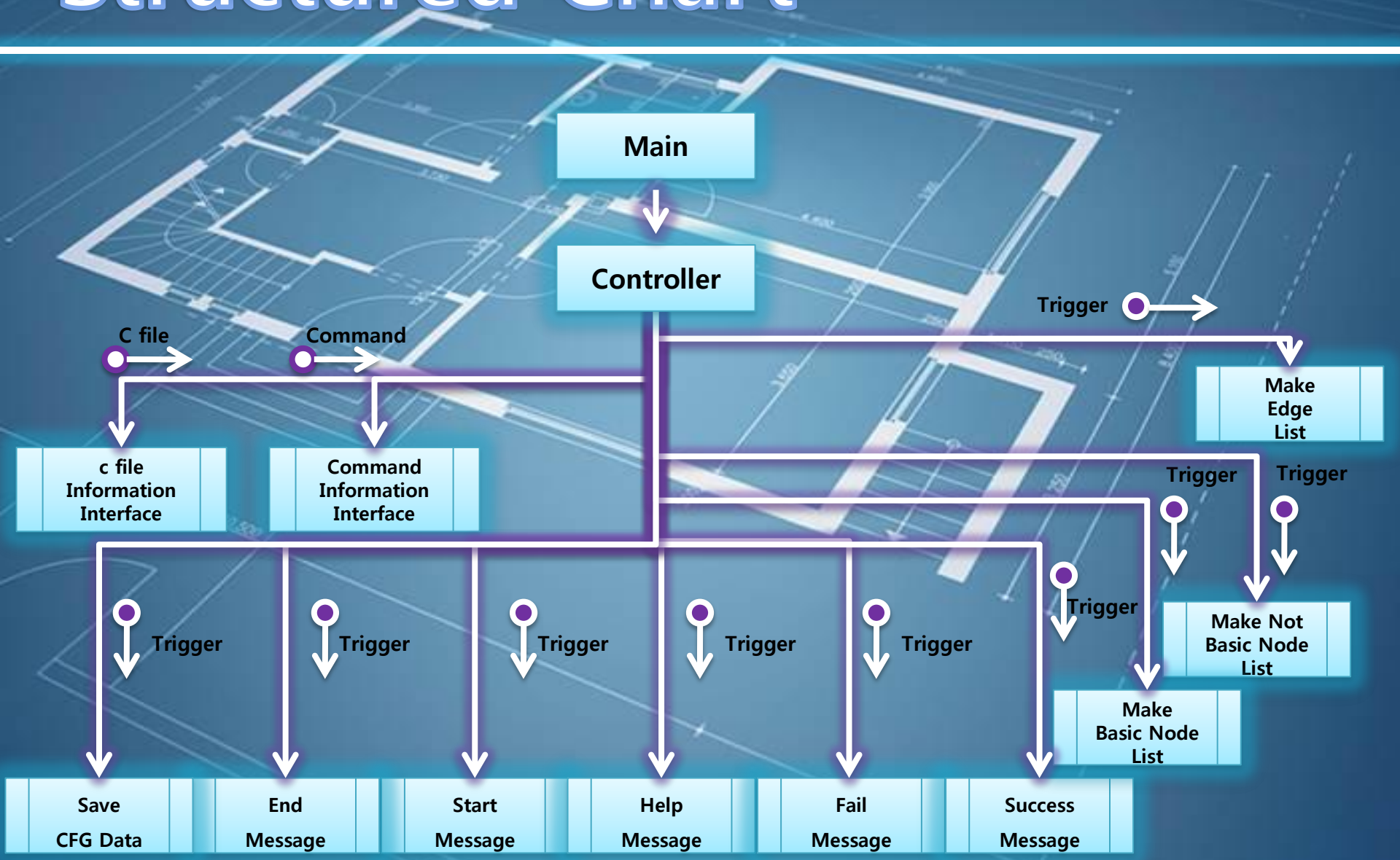
Structured Chart

Transform Analysis



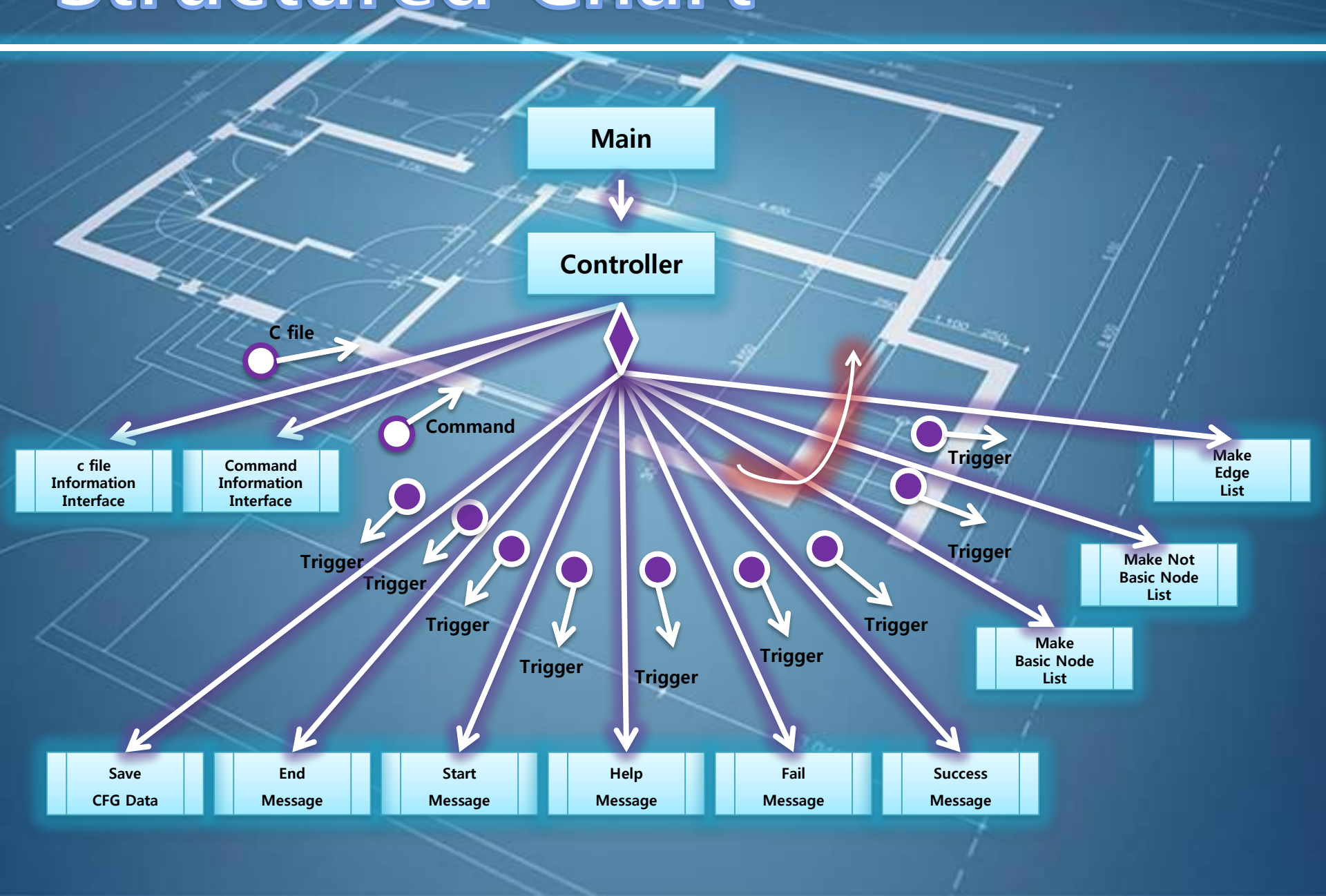
Structured Chart

Basic



Structured Chart

Advanced



Implementation

C file Information Interface

```
void C_file_Information_Interface()
{
        C_file_exp_checker(fn_in_c,fn_in_txt);
        C_file_exist_checker(fn_in_c);
        C_file_line_checker();
        C_file_header_checker();
        c_file_main_function_checker();
        message(sucess);
}
```

Reference No.	1.1
Name	c file Information Interface
Input	c file Input
Output	File Data
Process Description	After receiving c file Input, it sends File Data to CFG generator controller

Implementation

Command Information Interface

```
void Command_Information_Interface(int argc, char *argv[])
{

        if ( (argc < 3) || (argc > 3)){
        message(help);
        exit(1);
        }

        fn_in_c = argv[1];
        fn_in_txt = argv[2];

        printf("intput file name : %s₩n",fn_in_c);
        printf("result txt file : %s₩n",fn_in_txt);
}
```

Reference No.	1.2
Name	Command Information Interface
Input	Command Input
Output	Command Data
Process Description	After receiving Command Input, it sends Command Data to CFG generator controller

Implementation

Make Basic Node List

Make Not Basic Node List

```
if(status_m == 0 && status_main == 1)
                {
                                NODE_T *new_node;
                                new_node = (NODE_T*)malloc(sizeof(NODE_T));
                                NODE_T * p = head_node;
                                while(p->next != NULL){
                                p = p->next;
                };
                                new_node->next = p->next;
                                p->next =  new_node;

                                //new_node->data = (char *)
                                malloc(sizeof(char)*(strlen(a)+1));
                                //new_node->data = buffer;

                                int len = strlen(buffer);
                                if(buffer[len-1] = '₩n') buffer[--len] = '₩0';
                                strcpy(new_node->data, buffer);
                                new_node->line_number = node_number;
                                node_number++;
                                cnt++;
```

Reference No.	1.3.2 / 1.3.3
Name	Make Basic / Not Basic Node List
Input	Trigger
Output	CFG Command
Process Description	Create a list of Basic / Not Basic Node.

Implementation

Make Edge List

```
if((cnt>=1 || status_save !=1) && status_save_for != 1 && status_nosave !=1 )
                {
                                EDGE *new_edge;
                                new_edge = (EDGE*)malloc(sizeof(EDGE));
                                EDGE *o = head_edge;
                                while(o->next != NULL){
                                o = o->next;
                                };
                                new_edge->next = o->next;
                                o->next = new_edge;

                                if(status==0)
                                {
                                                new_edge->target = node_number - 1;
                                                new_edge->source = node_number - 2;
                                                new_edge->label = 0;
                                }
                                else if(status==1)
                                {
                                                new_edge->target = node_number - 1;
                                                new_edge->source = node_number - 2;
                                                new_edge->label = 0;
                                }
                                                ●
                                                ●
                                                ●
                }
```

Reference No.	1.3.4
Name	Make Edge List
Input	Trigger
Output	CFG Command
Process Description	Create a list of edges.

Implementation

Message Print Interface

```
void message(int a) {
                    switch(a) {

                  case help:
                    printf("#############################################################₩n");
                    printf("Help message: ./ex <input_c_file_name> <result_txt_file>₩n");
                    printf("#############################################################₩n");
                    break;

                  case fail:
                    printf("##############################₩n");
                    printf("File Not Open. Check C File.₩n");
                    printf("##############################₩n");
                    break;

                  case sucess:
                    printf("################₩n");
                    printf("File Open Sucess₩n");
                    printf("################₩n");
                    break;

                  case start:
                    printf("##################₩n");
                    printf("CFG Generate Start₩n");
                    printf("##################₩n");
                    break;
```

Reference No.	1.5
Name	Message Print Interface
Input	Message Command
Output	Message
Process Description	Display proper messages

Implementation

Print

CFG Data

```
while(cursor_edge != NULL)
    {
                cursor = head_node->next;
                while(cursor !=NULL)
                {
                        if(cursor_edge->source == -1)
                        {
                                strcpy(source_buff, "Start");
                        }
                        if(cursor->line_number == cursor_edge->source)
                        {
                                strcpy(source_buff, cursor->data);
                                //printf("%d₩n", cursor_edge->source);
                                //printf("%s₩n", source_buff);
                        }
                                if(cursor->line_number == cursor_edge->target)
                                strcpy(target_buff, cursor->data);
                                cursor = cursor->next;
                        }

                        printf("%s₩n", source_buff);
                        switch(cursor_edge->label){
                                case 0:
                                        printf(" -> ");
                                        break;
```

Reference No.	1.4
Name	CFG Print Interface
Input	CFG Command
Output	CFG List
Process Description	After receiving CFG Command, it sends CFG List to CFG Print in order to print CFG List

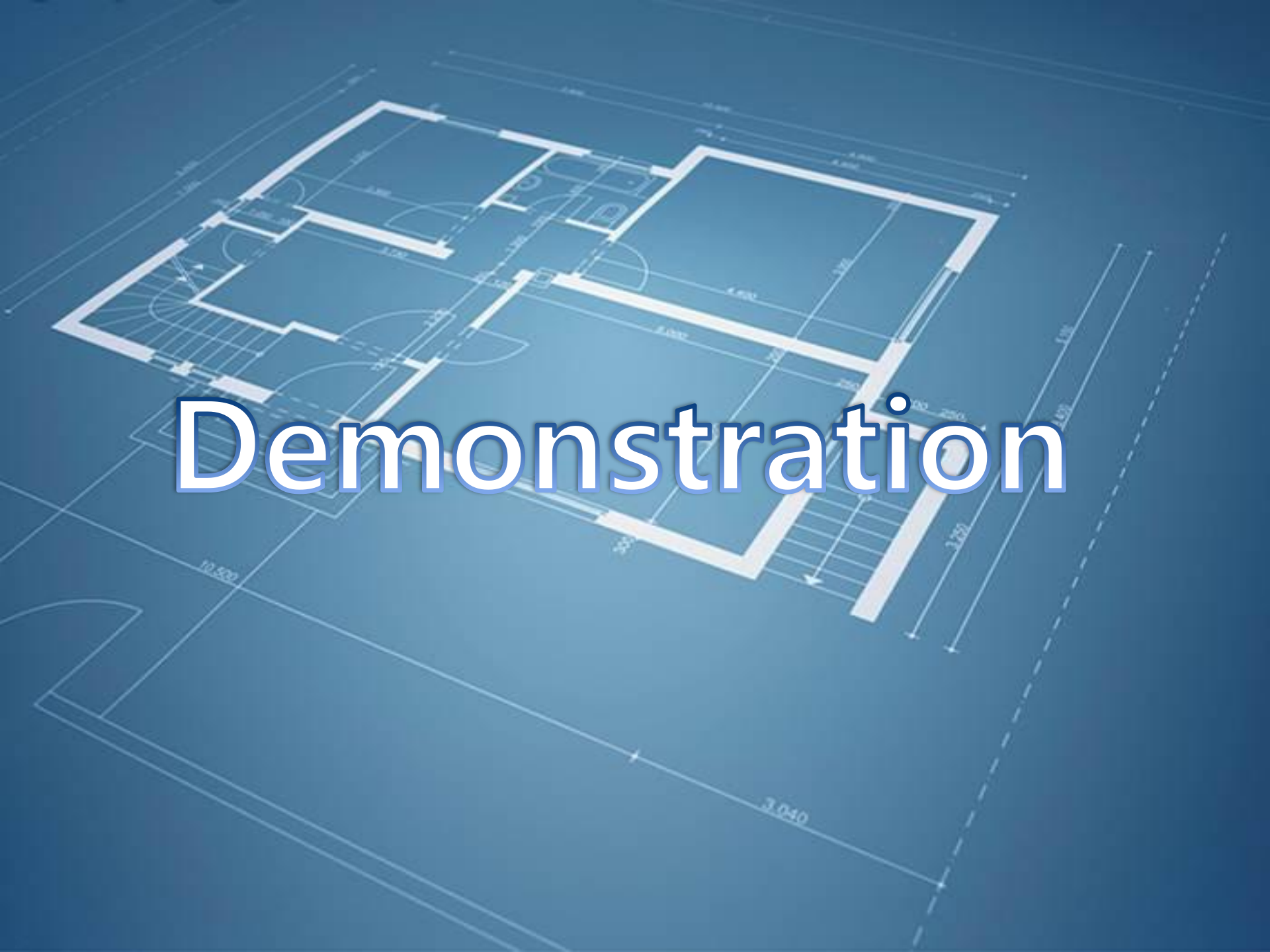
Implementation

Save

CFG Data

```
while(cursor_edge != NULL)
    {
            cursor = head_node->next;
            while(cursor !=NULL)
            {
                                    ●
                                    ●
                                    ●
                    fprintf(fp_save, "Source : %d  Target : %d label : %d ₩r₩n",
                    cursor_edge->source, cursor_edge->target, cursor_edge->label);
                    cursor_edge = cursor_edge->next;
            }
```

Reference No.	1.6
Name	CFG Save Interface
Input	Save Command
Output	CFG Data
Process Description	Transforms Save Command into CFG Data and saves it.


Demonstration


Q & A
alt