SASD of CFG Generator

Team 6
발표 : 200811425 김평석
200811435 신성호
200811451 이형열
200811454 전인서

Contents

Structured Analysis

Statement of Purpose

- Entire Statement of Purpose
- Specific Statement of Purpose

Entire Statement of Purpose

Entire Statement of Purpose
Program은 CUI(Command User Interface)기반의 Cygwin에서 구동된다.
C language Program을 입력 받아 CFG(Control Flow Graph)의 State목록과 Edge의 목록을 List형태로 출력한다.
C program의 유효성을 검사한 후 중간언어를 생성한다.
생성된 중간언어를 통하여 CFG정보를 도출한다.
도출된 CFG정보는 사용자 옵션에 의하여 File 과 Console에 각각 출력된다.
System Condition에 따라 Message를 Console에 출력한다.

Specific Statement of Purpose

Input Statement of Purpose
Command Line명령어 형태로 Source File Name, Output File Name과 Output Option을 입력 받는다.
입력되는 C Program File은 200줄 이내의 단일 파일이며 *.c의 확장자를 가져야 한다.
Main function을 반드시 포함하는 Code이어야 한다.
Pointer를 사용하지 않는 Code를 대상으로 한다.
사용자가 정의한 헤더와 Library를 사용한 파일에 대해서는 작동하지 않는다.
입력 받는 C Code파일은 문법적 오류가 없는 파일로 한정한다.

Specific Statement of Purpose

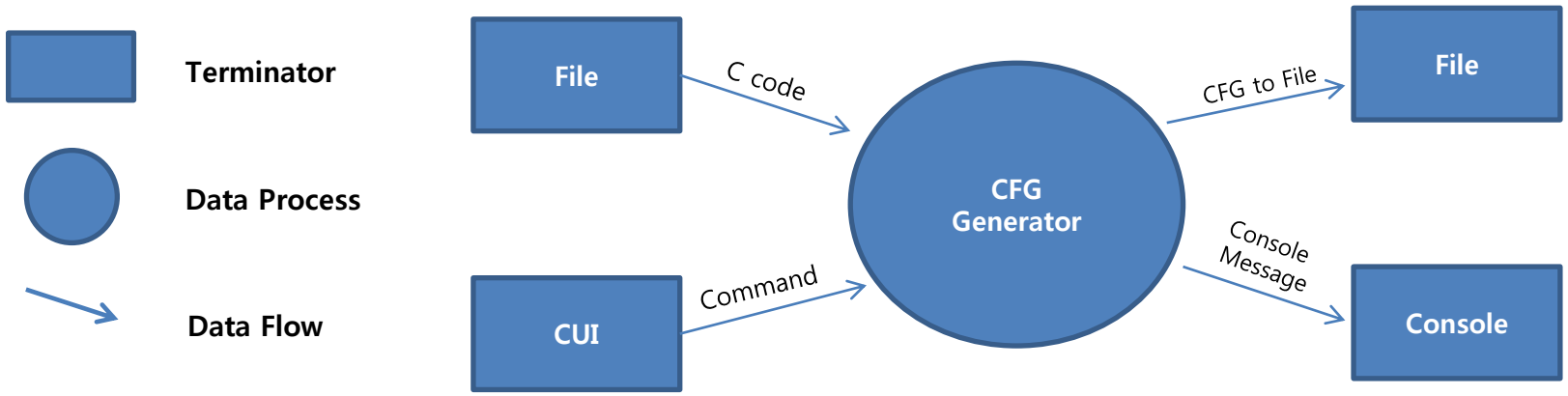
Parsing Statement of Purpose
입력 받은 C Code를 지정된 토큰을 기준으로 나눈다.
나눠진 부분의 동작을 파악하여 중간언어로 변환한다.
Macro에서 생성되는 분기는 고려하지 않는다.
삼항연산자(x ? y : z)는 분기를 생성하지 않는다.
Library와 Function안의 분기는 CFG에 나타내지 않는다.

Specific Statement of Purpose

Constructing CFG Statement of Purpose
중간언어를 분석하여 CFG의 Block List를 생성한다.
생성된 Block List를 이용하여 Edge List를 생성한다.

Output Statement of Purpose
Command Error 또는 File Error가 발생하면 Console에 Error Message와 Help Message를 출력한다.
출력옵션에 따라서 CFG component List를 이용하여 File과 Console에 각각 구조적으로 출력한다.

System Context Diagram & Event List



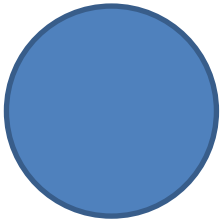
Input/Output Event	Description	Format/Type
Command	Input File Name, Output File Name 그리고 Output Option을 포함한 문자열 Ex) ./cfg [option] [input file name] [output file name]	string
C Code	CFG로 변환될 C code로 *.c의 확장자 명을 가지는 파일. C 표준에 의해 생성된 소스코드이다.	*.c
CFG to File	File에 출력될 CFG로서 string형식으로 파일에 쓰여지게 된다.	string
Console Message	콘솔에 출력할 메세지들과 CFG로 각 상태에 맞는 메세지들을 콘솔에 출력하도록 해준다.	CmdError / FileError / HelpMessage / GenerateMessage / CFG / OutFileName

Data Flow Diagram

Notation of Data Flow Diagram

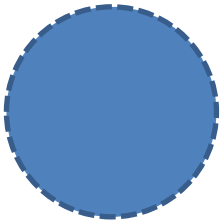
Terminator

Data Store



Data Process

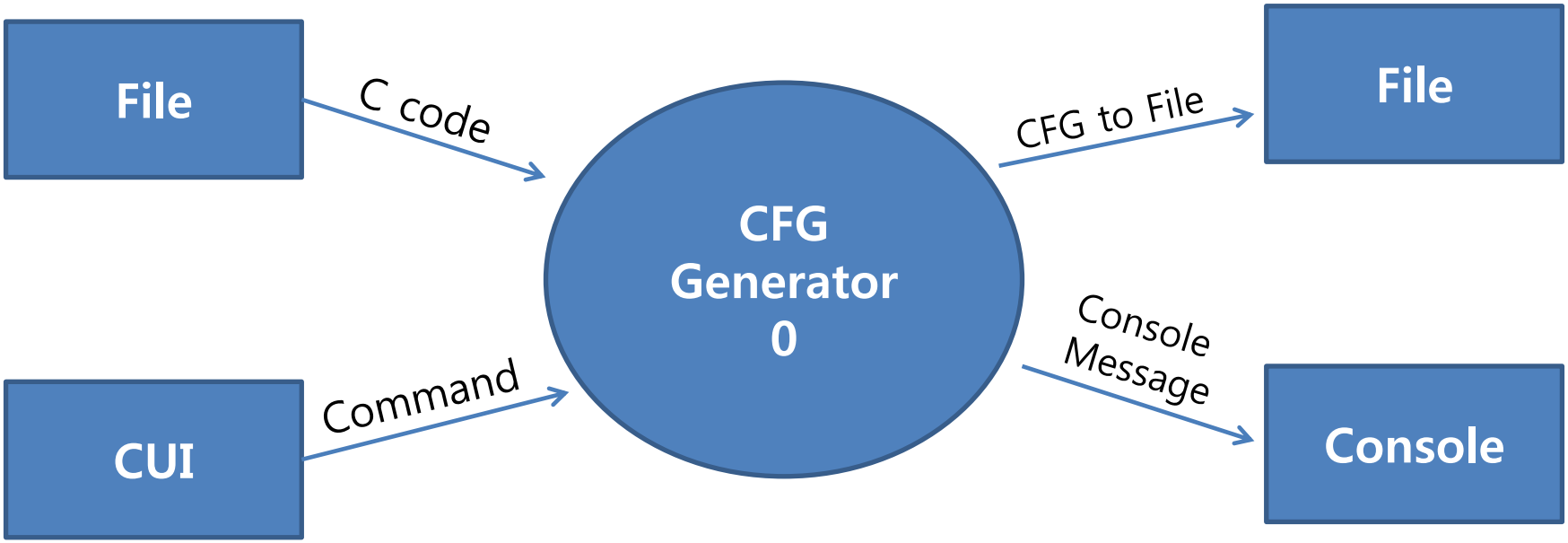
Data Flow



Control Process

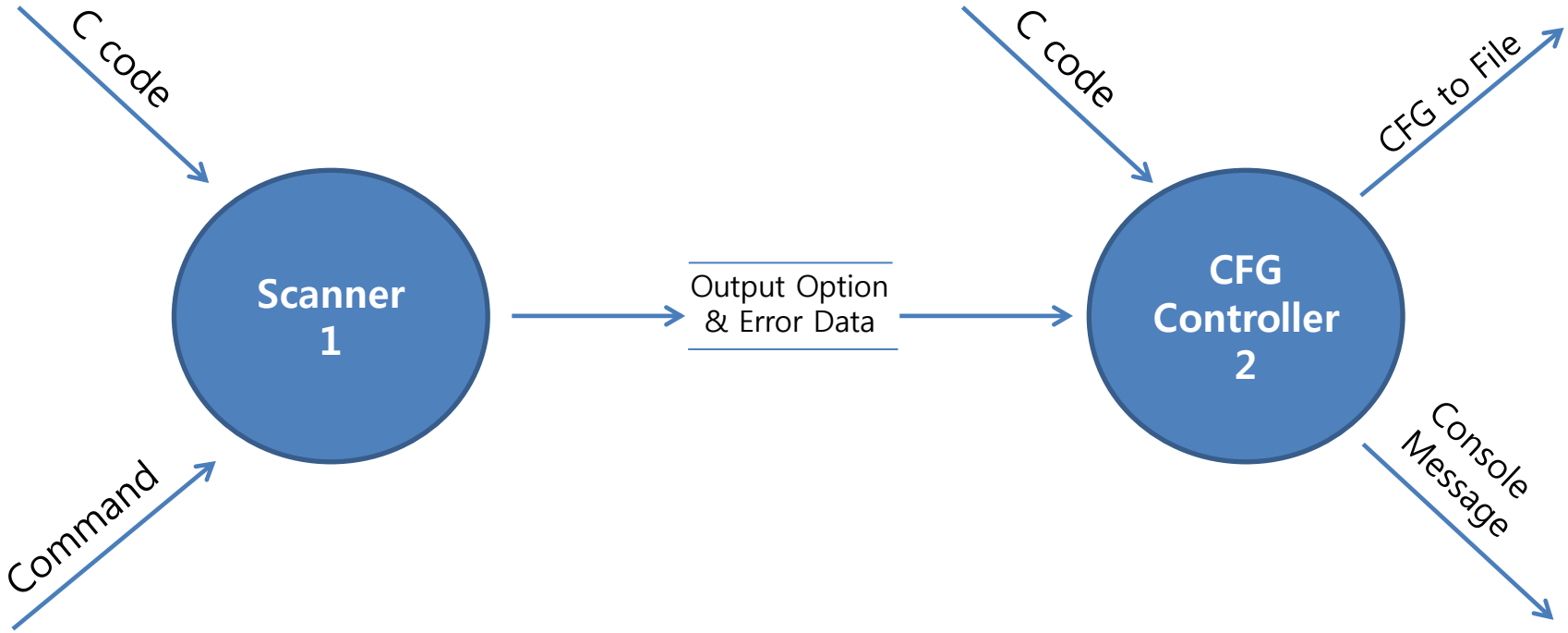
Control Flow

DFD Level 0



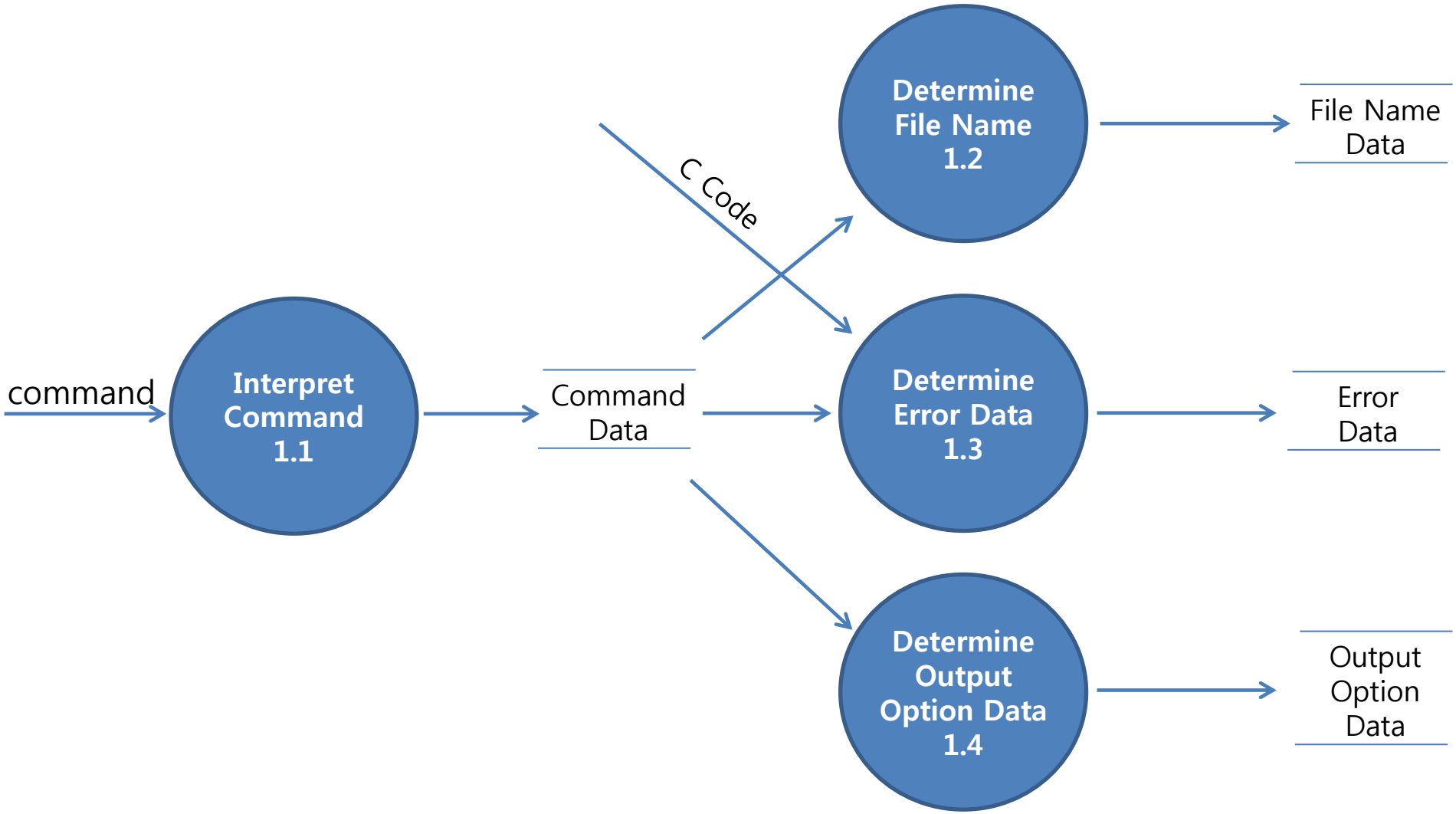
Data Dictionary

DFD Level 1



Data Dictionary

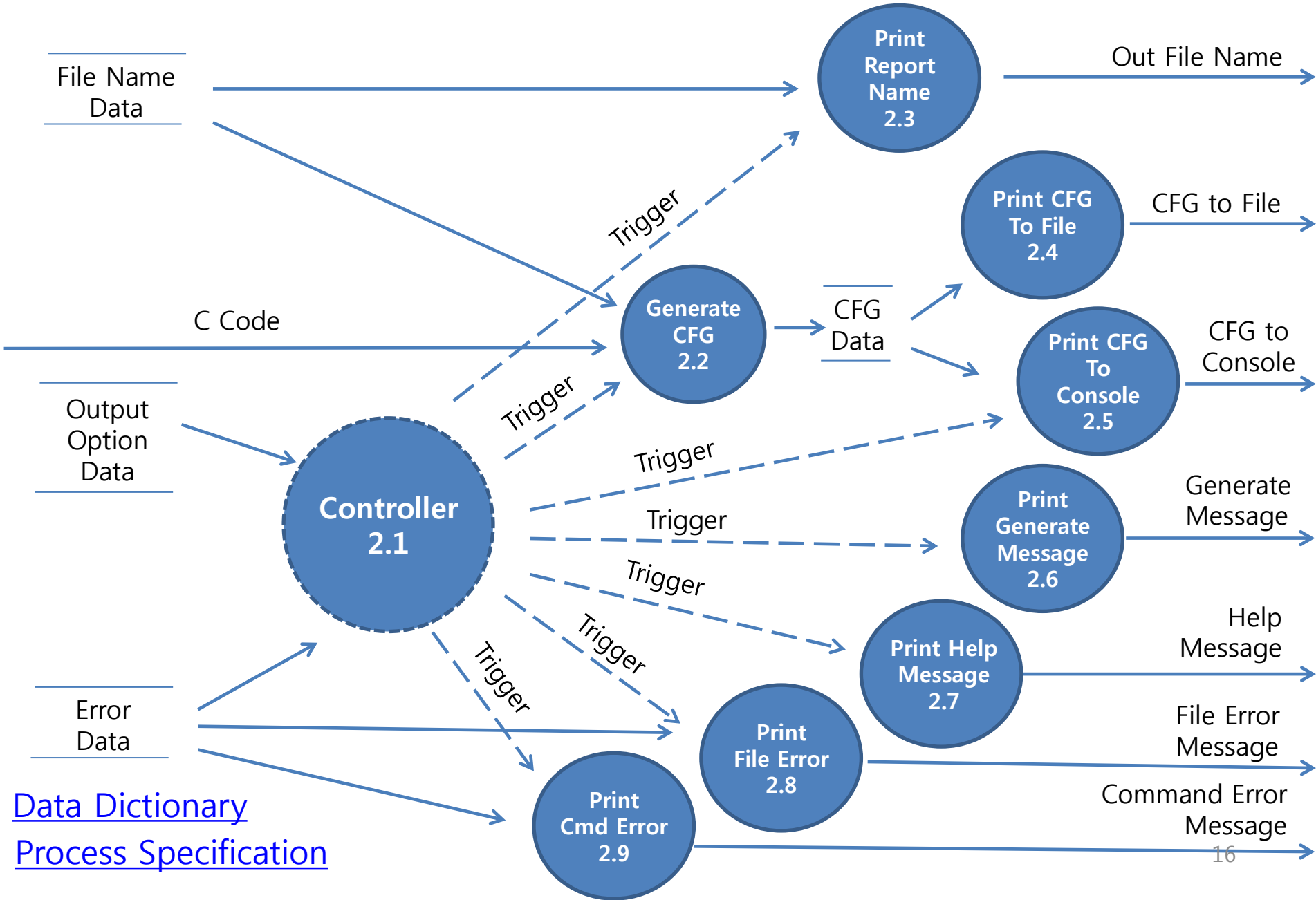
DFD Level 2



Data Dictionary

Process Specification

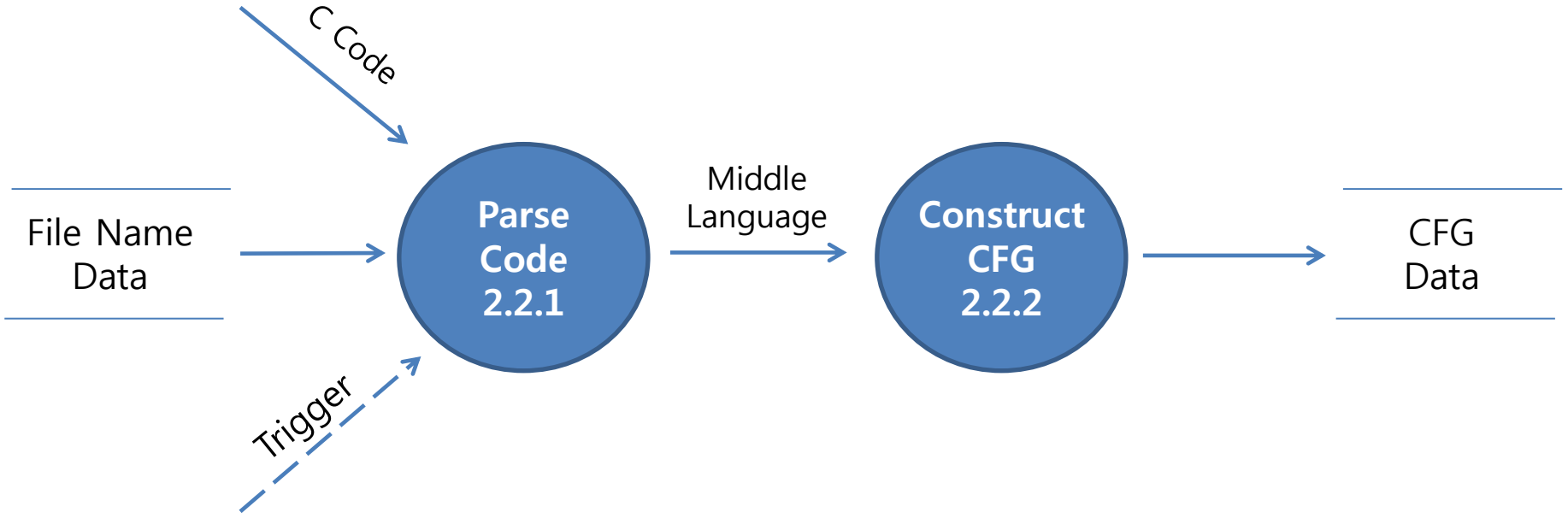
DFD Level 2



[Data Dictionary]

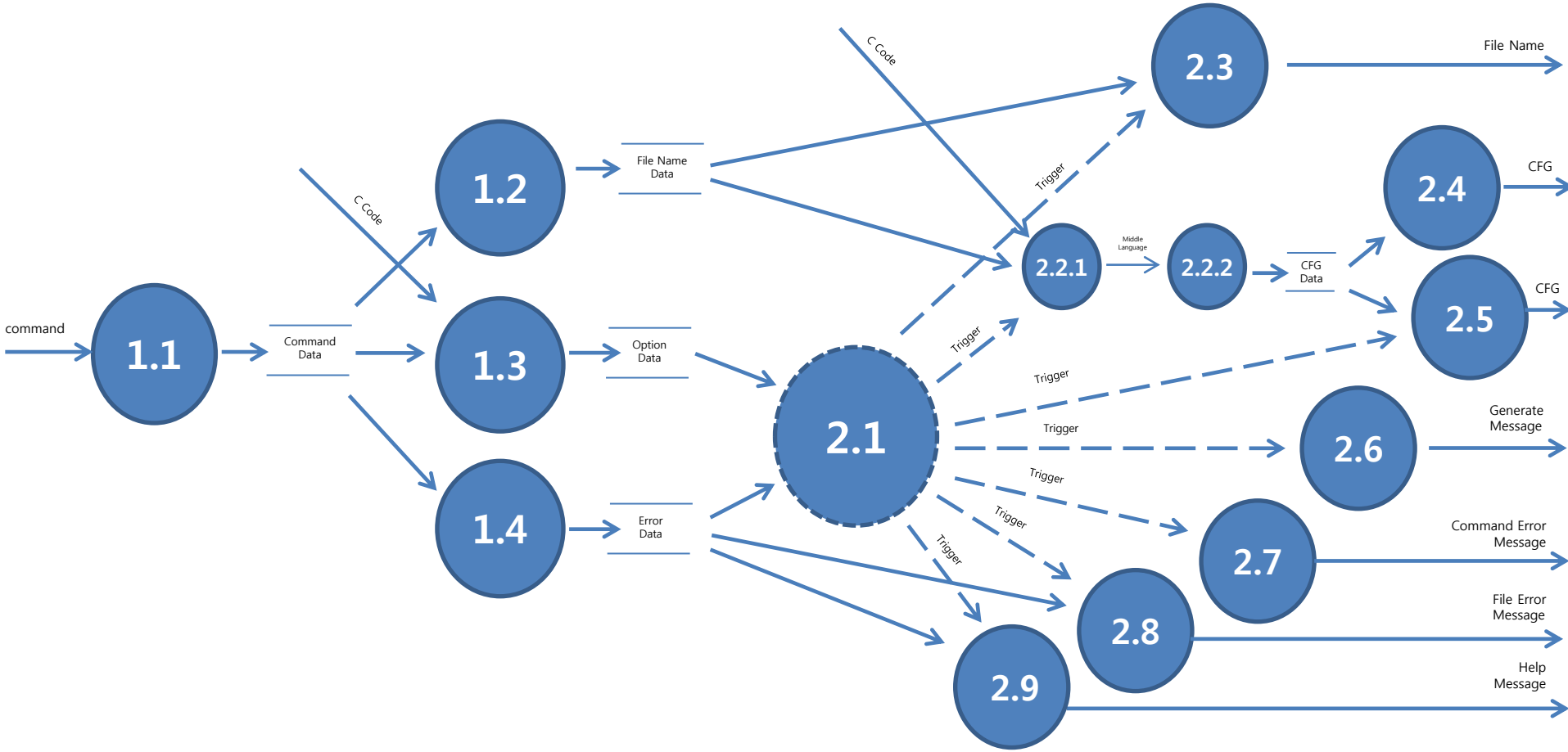
[Process Specification]

DFD Level 3



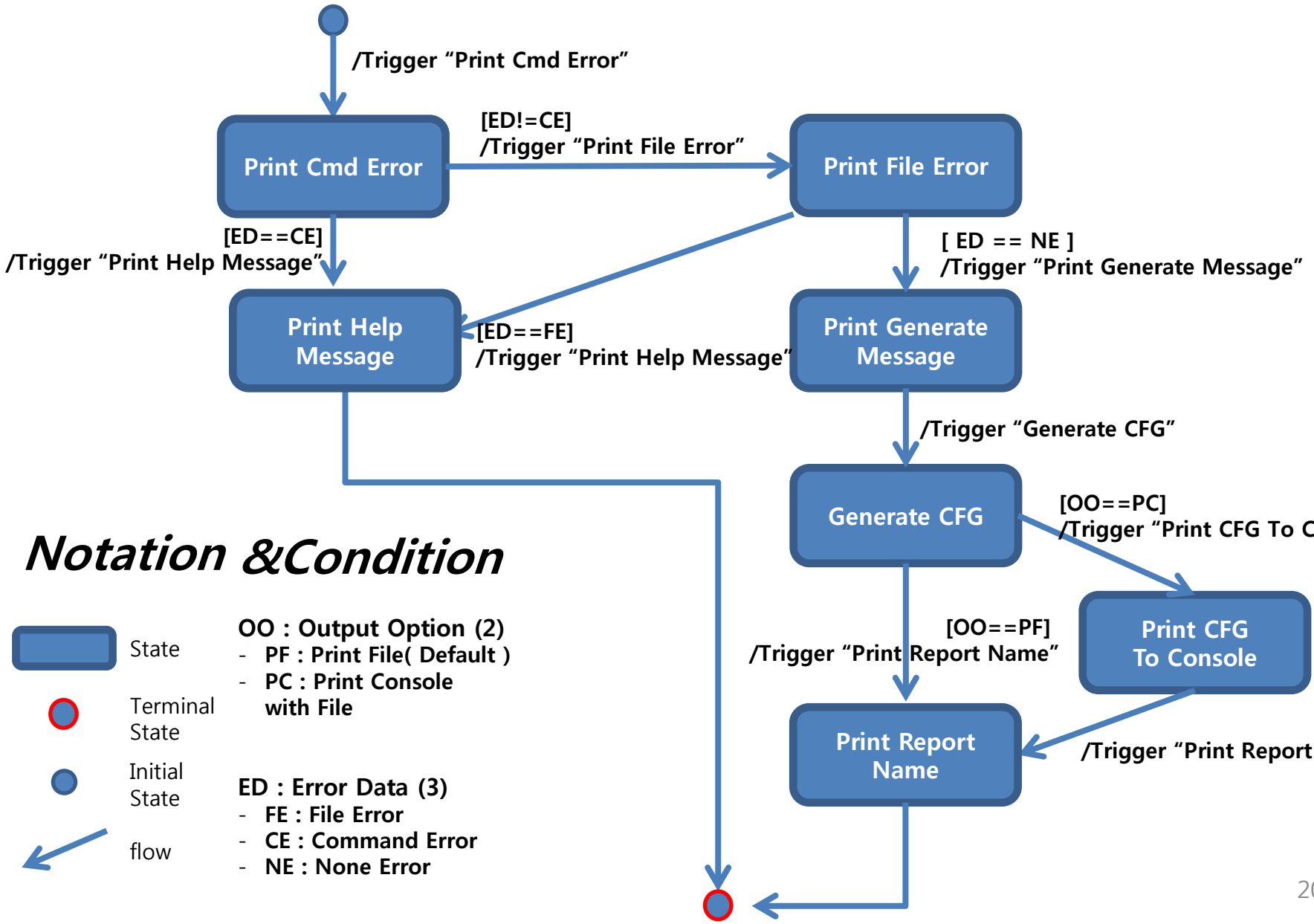
Process Specification

Data Flow Diagram



Final State Machine

Final State Machine



Notation &Condition

State

Terminal State

Initial State

flow

OO : Output Option (2)
- **PF : Print File(Default)**
- **PC : Print Console with File**

ED : Error Data (3)
- **FE : File Error**
- **CE : Command Error**
- **NE : None Error**

Data Dictionary

Data Dictionary Level 0

DFD

Input/Output Event	Description	Format/Type
Command	Input File Name, Output File Name 그리고 Output Option을 포함한 문자열 Ex) ./cfg [option] [input file name] [output file name]	string
C Code	CFG로 변환될 C code로 *.c의 확장자 명을 가지는 파일. C 표준에 의해 생성된 소스코드이다.	*.c
CFG to File	File에 출력될 CFG로서 string형식으로 파일에 쓰여지게 된다.	string
Console Message	콘솔에 출력할 메세지들과 CFG로 각 상태에 맞는 메세지들을 콘솔에 출력하도록 해준다.	CmdError / FileError / HelpMessage / GenerateMessage / CFG / OutFileName

Data Dictionary Level 1

DFD

Input/Output Event	Description	Format / Type
Output option & Error Data	Output Option과 Error Data를 저장 Error Data의 경우 ED 의 변수명을 갖는 string타입으로 저장되며, 각 상황에 따라 나뉜다. *CE (Command Error : command line으로 부터 입력받은 명령어의 문법& 형식), FE (File Error : 파일의 포맷과, 파일의 존재유무), NE (None Error : 에러가 없을경우)로 각각 표현되게 된다. Ex) string ED = “NE”; Output Option Data의 경우는 OO의 변수명을 갖는 string타입으로 저장되며, 각 상황에 따라 나뉜다. *PF(print only file), PC(print console with file)을 표현하게 된다. Ex string OO = “PC”;	string

Data Dictionary Level 2

DFD

Input/Output Event	Description	Format/Type
Command Data	Command Data를 저장한다. 입력 받은 명령어가 올바를 경우 미리 정해진 형태로 분석하여 struct형태로 Data들을 분리하여 저장하게 된다. Ex) ./cfg [option] [input file name] [output file name] 형태로 입력 받으면 각각 정보들을 string형태로 나누어 구조체에 저장한다. 만약 적절하지 않은 명령어가 들어오면 모두 NULL값이 저장된다.	Struct
File Name Data	Command Data의 정보를 이용하여 Input File Name과 Output File Name을 구조체에 저장한다	Struct

Data Dictionary Level 2

DFD

Input/Output Event	Description	Format/Type
Error Data	Command Data를 분석하여 명령어의 오류와 File Error를 체크하여 그에 상응하는 값을 string 형태로 저장하게 된다. ED 의 변수명을 갖는 string타입으로 저장되며, 각 상황에 따라 CE (Command Error : command line으로 부터 입력 받은 명령어의 문법& 형식), FE (File Error : 파일의 포맷과, 파일의 존재유무) NE (None Error : 에러가 없을 경우) 중 하나의 문자열을 저장한다. Ex) string ED = “NE”;	string
Output Option Data	Command Data로 부터 옵션을 받아와 OO의 변수명을 갖는 string타입으로 저장되며, 각 상황에 따라 PF(print only file), PC(print console with file)중 하나의 문자열을 저장한다. Ex) string OO = “PC”;	string

Data Dictionary Level 2

DFD

Input/Output Event	Description	Format/Type
CFG Data	CFG Data 를 저장한다. 입력된 C code로 부터 생성된 Data로 struct형태의 Node리스트 Edge리스트를 포함하는 struct로 생성되며, 이 Data를 토대로 File과 Console로 CFG를 출력하게 된다.	Struct
Command Error	Whether the cmd error occurred or not	String
Out File Name	Output file name that will convert to CFG and the file name that report will be saved in	string
File Error	Whether File exist or not	String
Help Message	*Help Message* that will outputs to Console	String
Generate Message	CFG generater's complete message	String
CFG (Console)	CFG that will outputs to console	string

Data Dictionary Level 3

DFD

Input/Output Event	Description	Format/Type
Middle Language	Parsing과정을 거쳐 생성된 Mid-Language 를 저장한다. 파싱과정에서 생긴 중간적인 언어로서 소스코드를 분석하여 CFG component를 생성하기 위한 Data이다. 각 분기별 일정 정보를 struct형태의 list또는 배열로 저장	Struct List / Array

Process Specification

Process Specification Level 2

DFD

Reference	1.1
Name	Interpret Command
Input	Command
Output	Command Data
Process Description	“Command” come in this Process from CUI. The data in the command consist of string is divided into three part of string by this process, “Interpret Command”. After that work, this process save Command Data Systematically .

Process Specification Level 2

DFD

Reference	1.2
Name	Determine File name
Input	Command Data
Output	File Name Data
Process Description	“Command Data” come in this Process from The storage. The data have “File Name Data” that is consist of Report Name(Output File Name) and C Code File. This Process extract the data from “Command Data” And save File Name data Systematically for convenience of Controller.

Process Specification Level 2

DFD

Reference	1.3
Name	Determine Error Data
Input	Command Data
Output	Error Data
Process Description	“Command Data” come in this Process from The storage. The data have “Error Data” that is consist of File Error Data and Cmd Error Data. This Process try to open the C Code File to check the File’s validity. And save Error data Systematically form convenience of Controller.

Process Specification Level 2

DFD

Reference	1.4
Name	Determine Output Option Data
Input	Command Data
Output	Output Option Data
Process Description	“Command Data” come in this Process from The storage. The data have “Output Option Data” that have two condition, Print File and Print Console. The Print File is default option when any option does not income. After extract Output Option Data from Command Data, This process save output option data for convenience of Controller.

Process Specification Level 2

DFD

Reference	2.1
Name	Controller
Input	Error Data, Output Option Data
Output	Trigger
Process Description	“Error Data” and “Output Option data” come in this process for divergence. Controller can make a decision with that data for process of Program. The Operation of this Process is represented to “state machine” specifically.

Process Specification Level 2

DFD

Reference	2.3
Name	Print Report Name
Input	Trigger, File Name Data
Output	File Name
Process Description	After Triggered, this process print report name as meaning of finis hing print CFG and succeeding this program. Since this process can't take report name to controller, it take report name to File Name Data

Process Specification Level 2

DFD

Reference	2.4
Name	Print CFG To File
Input	Trigger, CFG Data
Output	CFG
Process Description	After Triggered, this process, “Print CFG To File” print CFG Data to File. Since CFG go out two stream, File and Console, this process is named ‘Print CFG To File’ not ‘Print CFG’.

Process Specification Level 2

DFD

Reference	2.5
Name	Print CFG To Console
Input	Trigger , CFG Data
Output	CFG
Process Description	After Triggered, this process ,'Print CFG To Console', print CFG Data to Console. This process operate selectively, the contrary "Print CFG To File" operate unconditionally after generate CFG.

Process Specification Level 2

DFD

Reference	2.6
Name	Print Generate Message
Input	Trigger
Output	Generate Message
Process Description	After Triggered, this process print Generate Message meaning of ‘No Error’ and starting ‘Generate CFG Data’

Process Specification Level 2

DFD

Reference	2.7
Name	Print Cmd Error
Input	Trigger
Output	Command Error Message
Process Description	When Cmd Error occur, The controller trigger this process to print command error message. After print command error message, controller trigger 'print help' process . If there is no error in command, this process does not print error message.

Process Specification Level 2

DFD

Reference	2.8
Name	Print File Error
Input	Trigger
Output	File Error Message
Process Description	When File Error occur, The controller trigger this process to print file error message. After print command error message, controller trigger ‘print help’ process. If there is no error in command, this process does not print error message.

Process Specification Level 2

DFD

Reference	2.9
Name	Print Help Message
Input	Trigger
Output	Help Message
Process Description	Whenever Error occur, this process is triggered. This process help user to operate ‘CFG Generator’ correctly. After print help message, this program is terminated.

Process Specification Level 3

DFD

Reference	2.2.1
Name	Parse Code
Input	Trigger, File Name data, C Code
Output	Middle Language
Process Description	When Error isn't occurred, this process read 'C code' from File. This process translates C code to middle language, reading each line of c code.

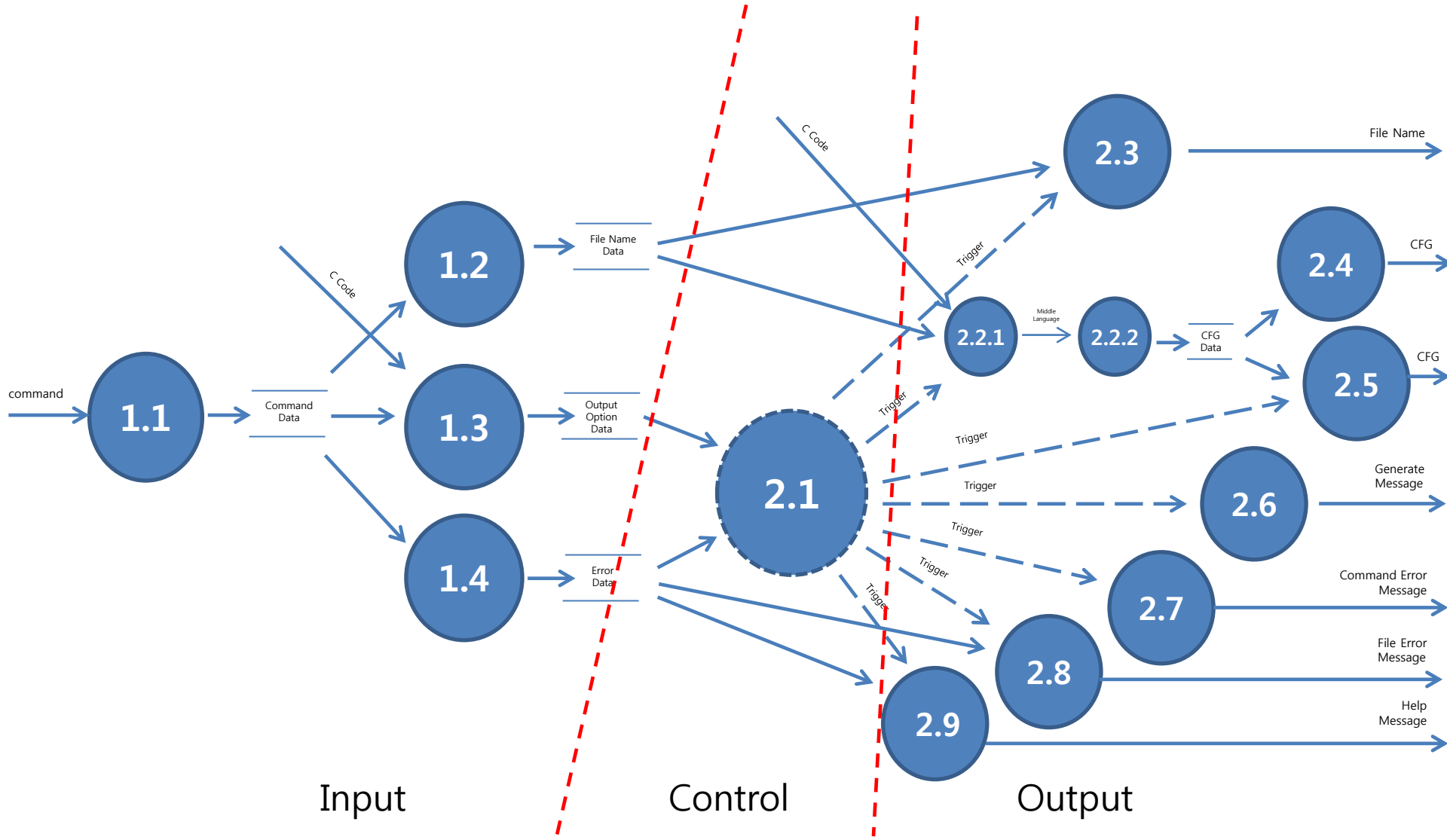
Process Specification Level 3

DFD

Reference	2.2.2
Name	Construct CFG
Input	Middle Language
Output	CFG Data
Process Description	After Middle Language is generated, This process construct CFG with middle language. Finishing the constructing, this process save CFG Data for print report and console.

Structured Design
- Structured Charts

Transform Analysis



Notation

Modules

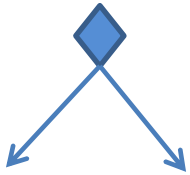
Library Modules

Data module

Module call

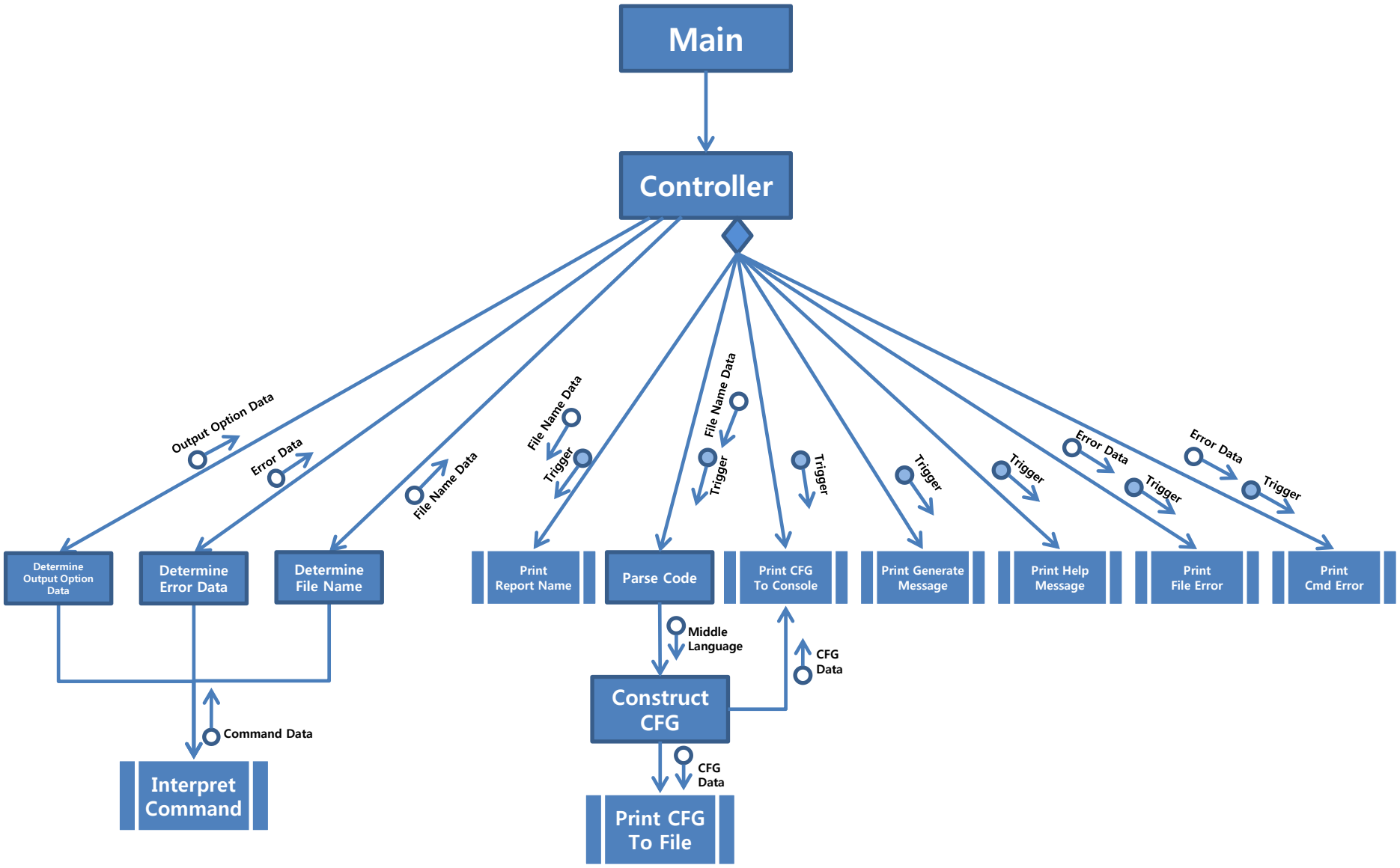
Data Flow

Control Flow

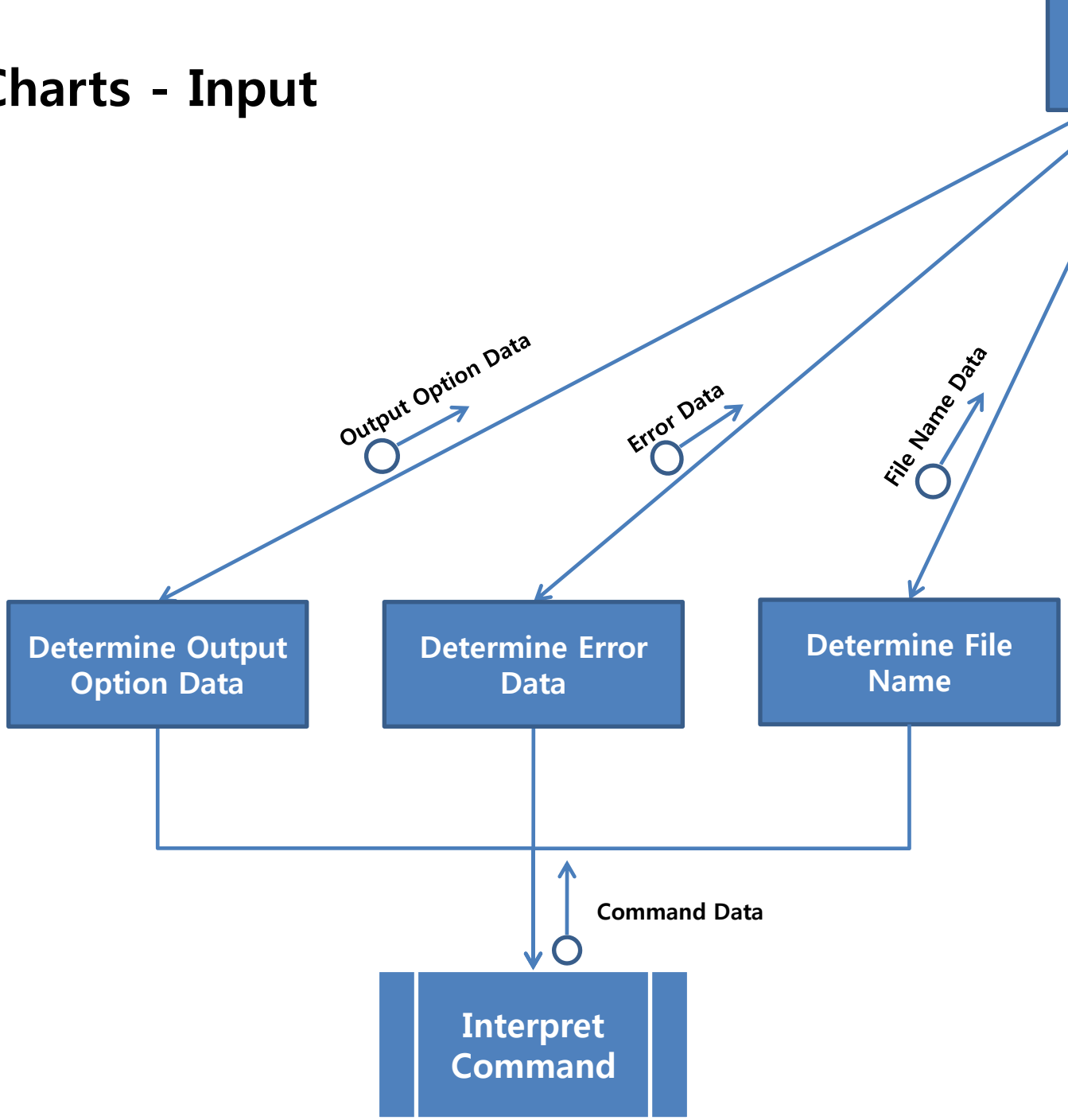


Decision

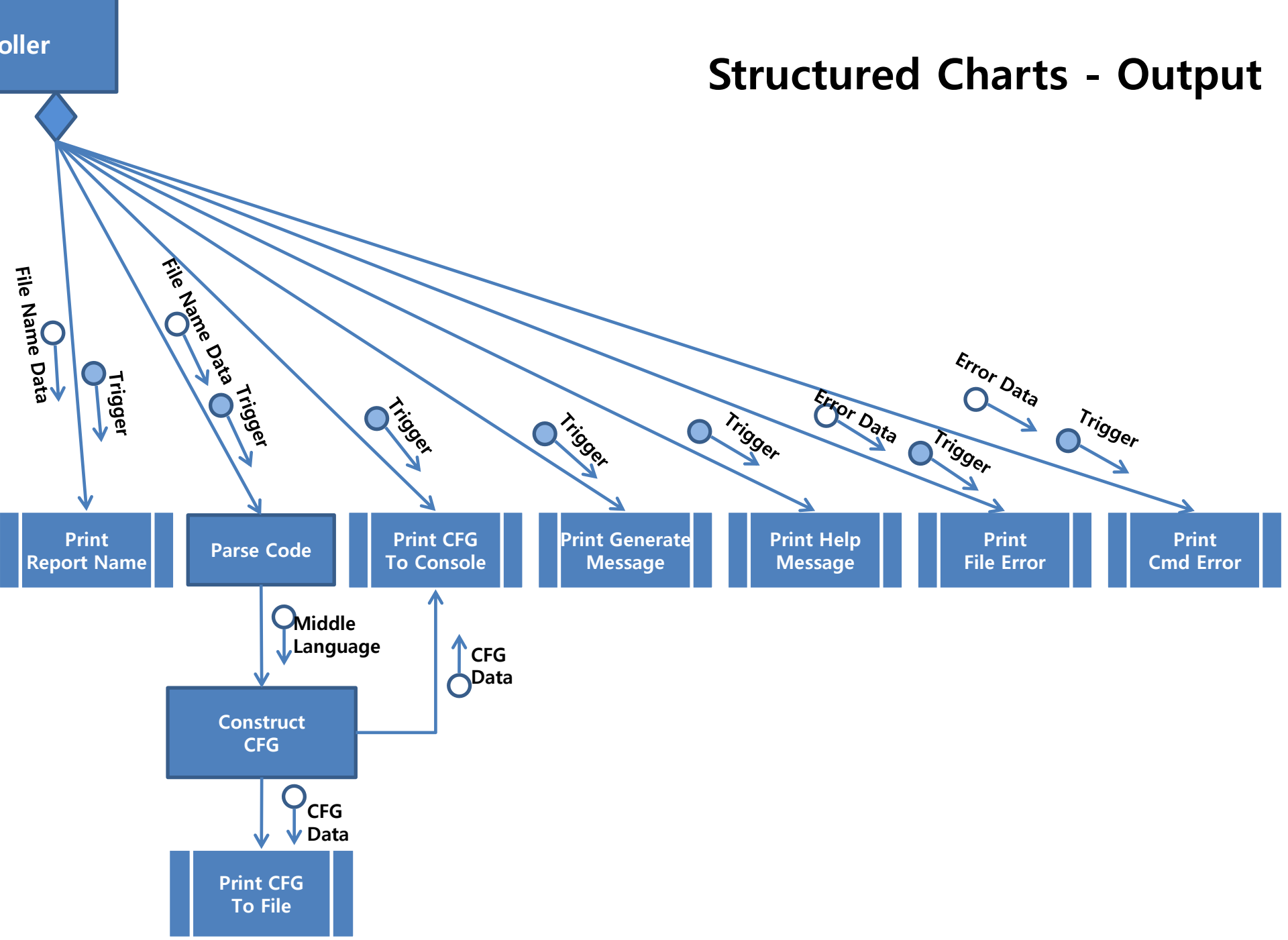
Structured Charts



Structured Charts - Input



Structured Charts - Output



Appendix

- Middle Language
- Parse Code
- Construct CFG

Middle Language

Middle Language

중간언어는 코드를 읽어서 CFG로 출력하기 좋은 형태로 변환 시킨 언어이다.
중간언어는 다음과 같은 이유로 생성하여 사용한다.

1. C코드에서 CFG로 변환 시에 중복될 수 있는 것들을 하나로 변환시켜준다.
   ex) for문이든 while문이든 CFG로는 동일한 구조를 갖는다. 이러한 것을 while for 따로 직접 처리해 주는 것이 아닌 중간언어로 동일한 코드로 만들어서 처리해준다.

2. CFG를 코드상태로 구체적으로 표시해준다.
   ex) 구체적으로 어디서 어디로 점프하는지 명시해 줌으로써 CFG에서 조금 더 흐름을 보기 좋게 한다.

3. 구현 시에 모듈화 및 분업에 유용하다.
   ex) 중간언어를 설정하여 Parse Code에서는 중간언어를 생성하는 부분을 만들고, Construct CFG 에서는 중간언어를 CFG로 만듦으로써 구현 시에 이 부분에서 한 사람이 구현을 하기 위해서 프로젝트 딜레이 되는 것을 줄여준다.

Parse Code

Parse Code
파서는 C코드를 읽으면서 다음과 같이 중간언어를 생성한다. 1. C코드에서 분기를 드는 지점을 파악하고 분기[조건]을 써준다. 2. 각각의 분기에 넘버링 해주고, 각각을 넘버링에 따라서 나열한다. 3. 넘버링된 분기에 각각 해당되는 [바디]를 채워 넣는다.

Middle Language

Middle Language - Keyword

중간언어의 키워드는 다음과 같이 이루어져 있다.

n#COND $condition$ -> 조건을 검사하여 분기를 만드는 블럭을 명시

n#BODY $code$ -> 조건 및 분기가 없는 일반적인 블럭을 명시

#JUMP n -> 블럭의 순서 및 관계를 명시

** $사이에는 코드가 랩핑되어서 중간언어와 구분된다.

** n 각 블럭에 대한 넘버링

** # 키워드에 앞서서 키워드를 명시해줌

Middle Language

Middle Language – 분기의 예

```
*** while(조건) { 코드 }
3#COND $조건$
#JUMP $6$          // 조건이 만족되지 않았을 때의 연결
4#BODY $코드$      // 코드를 실행하고
#JUMP $3$          // 다시 조건을 검사한다.
6#BODY $코드$      // while문 밖의 연결
……
```

```
*** do{ 바디 } while ( 조건 )
11#BODY $코드$
12#COND $조건$    // 조건검사를 해서
#JUMP $11$         // 조건이 만족 되었을 때 다시 11로 올라간다.
13#BODY $코드$    // do while 밖에서의 코드
….
```

Middle Language

Middle Language – 분기의 예

```
*** if(조건){코드} else if(조건){코드} else{코드}
25#COND $조건$
26#BODY  $코드$
#JUMP $30$
27#COND $조건$
28#BODY  $코드$
#JUMP $30$
29#BODY  $코드$
30#BODY  $코드$
…
```

Middle Language

Middle Language – 분기의 예

```
switch(변수) case 상태1: 코드 break; case 상태2:  코드 case 상태3: ....
25#COND $상태1$      //변수가 상태에 만족하면
26#BODY  $코드$
#JUMP $31$           // break가 있기 때문에 switch를 탈출한다
27#COND $조건$
28#BODY  $코드$      // break가 없기 때문에 바로 다음 조건도 실행한다.
29#COND $조건$
30#BODY  $코드$
31#BODY  $코드$      // switch 밖의 코드
...
```

```
for(초기문;조건;증감){바디}
43#COND$조건$
44#BODY  $코드$
#JUMP $43$
45#BODY  $코드$
...
```

Construct CFG

Construct CFG

Construct CFG는 위에서 생성된 중간언어를 분석하여 다음과 같은 형태로 출력해준다.

```
1#
코드....
2#
조건 .....
    3#
    코드..
4#
코드.....
5#
조건.....
    6#
    조건....
        7#
        코드....
    8#
    코드....
9#
코드.....
..
```

Thank Yoo and us