

# OSP Stage 1000

## <Plan & Elaboration>

### 그놈! Clone Checker

Project Team

T4

Date

2016-05-31

---

Team Information

201411258 강태준

201411265 김서우

201411321 홍유리

# Contents

**Activity1001. Define Draft Plan**

**Activity1002. Create Preliminary Investigation Report**

**Activity1003. Define Requirements**

**Activity1004. Record Terms in Glossary**

**Activity1005. Implement Prototype**

**Activity1006. Define Business Use Case**

**Activity1007. Define Business Concept Model**

**Activity1008. Define Draft System Architecture**

**Activity1009. Define System Test Case**

**Activity1010. Refine Plan**

# Activity1001. Define Draft Plan

## 1. Motivation

여태껏 2년동안 학기 중에 크고 작은 실습 과제를 하면서 서로의 코드를 복사하여 제출하는 학생들이 많이 있었다. 물론 우리 팀원들도 1학년 때는 그 방법을 많이 사용하였었다. 하지만 2학년 2학기가 끝나고 3학년 1학기를 시작하는 지금에서야 든 생각은 컴퓨터공학과 학생으로써 코드를 직접 구현하지 않고 다른 사람의 코드를 그대로 가져다 쓰는 것은 자신의 실력 향상에 도움이 전혀 안될 뿐 더러, 다른 사람의 노력을 헛되게 하는 행동이라는 것이다. 따라서, 컴퓨터공학과 후배들이 우리와 같은 뒤늦은 후회를 하지 않고 지금부터라도 코드 직접 구현 능력을 향상 시켰으면 좋겠다는 바람에서 코드의 유사성을 체크하는 프로그램의 개발을 하고자 한다.

## 2. Project Scope

다른 사람의 코드를 가져다 쓰는 사람들이 많이 있다. 컴퓨터공학과 학생으로써 코드를 직접 구현하지 않고 다른 사람의 코드를 그대로 가져다 쓰는 것은 실력 향상에 문제가 되기 때문에 하지 않는 것 이 좋은 행동이기에 그런 행동들을 하지 못하도록 Clone Checker 를 만들어서, 검사를 하도록 한다.

## 3. Project Objectives

실제 컴퓨터공학과 학생들의 과제 제출 시에 자신이 직접 구현하지 않고 다른 학생의 코드를 복사하여 낼 때 사용하는 방법들을 조사한 뒤, 그것을 기준으로 세워서 두 프로그램 코드의 유사도를 검사하는 프로그램이다. 각 기준 마다 가중치를 두어 프로그램을 실행하면 유사도를 비율(%)로 사용자에게 알려주는 것을 목표로 한다.

## 4. Functional Requirements

- Display Main
- Input Path

- Setting Files
- Start Analyze Code
- Analyze Variable
- Analyze Loop
- Analyze Conditional
- Analyze Function
- Make Detail
- Find X\_File
- Show X\_File
- Show Detail
- Exit

## 5. Non-Functional Requirements

- A. 50 개의 파일을 1 분 이내로 검사하고 원본으로 추정되는 파일을 찾는다

## 6. Resource Estimation

- A. Human Efforts ( Man - Month ) : 3 - 3
- B. Human Resource : 컴퓨터 공학 전공 학과생 3명
- C. Project Duration : 16주
- D. Cost : 100만원 (식대)

## Activity1002. Create Preliminary Investigation Report

### 1. Alternative Solutions

- A. 개발 전문 업체에 의뢰하여 제작한다.
- B. 기존의 프로그램에 사용된 알고리즘을 이용한다.

### 2. Project Justification (Business Demands)

- A. Cost : 100 만원
- B. Duration : 16 주
- C. Risk : OSP 경험 부족, JAVA 언어 이해 부족, UML 사용 경험 부족, 학생회 활동, 타 과목의 과제, 시험 기간
- D. Effect : C 프로그램 사이의 코드 유사도 확인

### 3. Risk Management

Risk	Probability	Significance	Weight
OSP 경험 부족	5	5	25
JAVA 이해 부족	4	3	12
UML 경험 부족	4	3	12
학생회 활동	3	4	12
타 과목 과제	2	3	6
시험 기간	4	5	20

#### 4. Risk Reduction Plan

Risk	Reduction Plan
OSP 경험 부족	지난번 동일 강의의 자료를 참고하거나 교수님, 조교, 선배들에게 자문을 구하여 도움을 얻는다.
JAVA 이해 부족	Java 관련 서적 및 관련 사이트 글을 읽고 도움을 얻는다.
UML 경험 부족	UML 관련 서적 및 관련 사이트 글을 읽고 도움을 얻는다.
학생회 활동	해야 되는 일이 생기면 빠르게 처리하고 학생회원들과 협력한다.
타 과목 과제	과제가 나오는 날 한다.
시험 기간	조원들끼리 협력하여 서로의 도움을 주고 받는다.

#### 5. Market Analysis

이미 코드 유사성을 위한 알고리즘 분석 논문은 많지만, S/W 제품은 그다지 많지 않다. 또한 기존의 제품은 우리 학교 우리 학과 학생들이 다루기 어려운 제품이 많다.

#### 6. Other Managerial Issues

2016년 6월까지 개발이 완료되어야 한다.

## Activity1003. Define Requirements

### 1. Functional Requirements

Function	Description
Display Main	UI 를 포함한 실행 초기 화면
Input Path	검사하는 파일들이 저장되어 있는 폴더의 경로를 입력해준다.
Setting Files	분석에 앞서 파일을 분석하기 쉽게 정리한다
Start Analyze Code	분석을 시작한다.
Analyze Variable	변수에 대한 검사를 한다.
Analyze Loop	반복문을 검사한다
Analyze Conditional	조건문을 검사한다
Analyze Function	함수에 대한 검사를 한다
Make Detail	최종 유사도 점수를 계산하고, 검사의 세부 내용을 .txt 파일에 저장한다.
Find X_File	X_File을 찾는다.
Show X_File	X_File의 파일명을 보여준다
Show Detail	검사의 세부 내용을 보여준다.
Exit	유사도 검사 프로그램을 종료한다.

Ref. #	Function	Category
R 1.1	Display Main	Evident
R 1.2	Input Path	Evident
R 1.2.1	Setting Files	Hidden
R 2.1	Start Analyze Code	Evident
R 2.1.1	Analyze Variable	Hidden
R 2.1.2	Analyze Loop	Hidden
R 2.1.3	Analyze Conditional	Hidden
R 2.1.4	Analyze Function	Hidden
R 2.1.5	Make Detail	Hidden
R 2.1.6	Find X_File	Hidden
R 3.1	Show X_File	Evident
R 3.2	Show Detail	Evident
R 4.1	Exit	Evident

## 2. Operating Environments

OS : Windows 7

IDE : Eclipse Mars2 Release(4.5.2)

개발 언어 : JAVA 1.8.0\_77

## 3. Development Environments

OS : Windows 7

IDE : Eclipse Mars2 Release(4.5.2)

개발 언어 : JAVA 1.8.0\_77

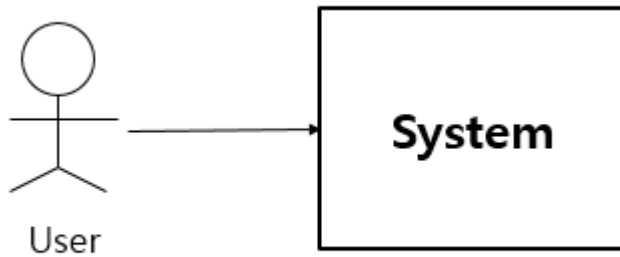
## Activity1004. Record Terms in Glossary

Term	Description	Remarks
Display Main	프로그램 시작 후 초기화면	
Input	입력한다	
Path	경로	
Setting	검사 준비	
Files	검사 할 파일들	
Start	시작하다	
Analyze	분석하다	
Variable	변수	
Loop	반복문	
Conditional	조건문	
Function	함수	
Draw	도출하다	
Show	보여주다	
Find	찾는다	
X_File	그놈 파일	
Detail	분석 결과 세부내용 파일(.txt)	
Exit	종료	



## Activity1006. Define Business Use Case

### 1. Define System Boundary



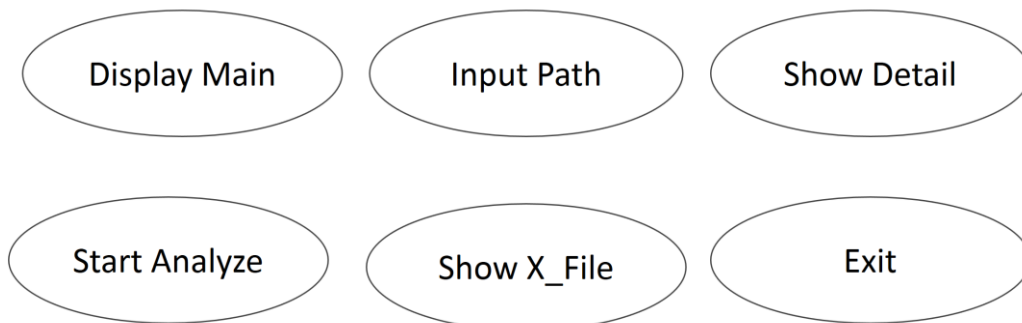
### 2. Identify and Describe Actors

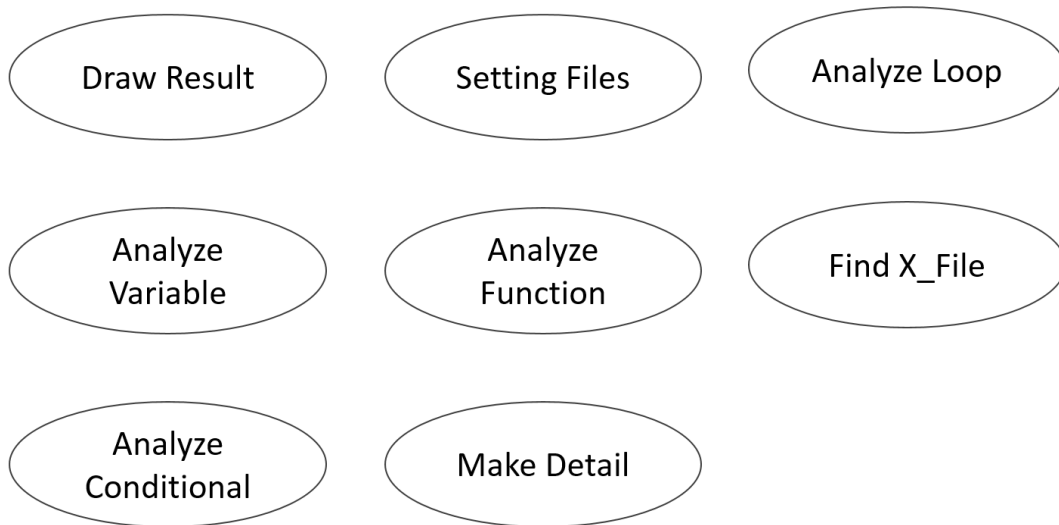
#### A. Actor

두 가지 코드의 유사도를 비교하기 위하여 본 프로그램을 사용하는 컴퓨터공학과 학생,  
조교

### 3. Identify Use-Case

#### A. Actor based



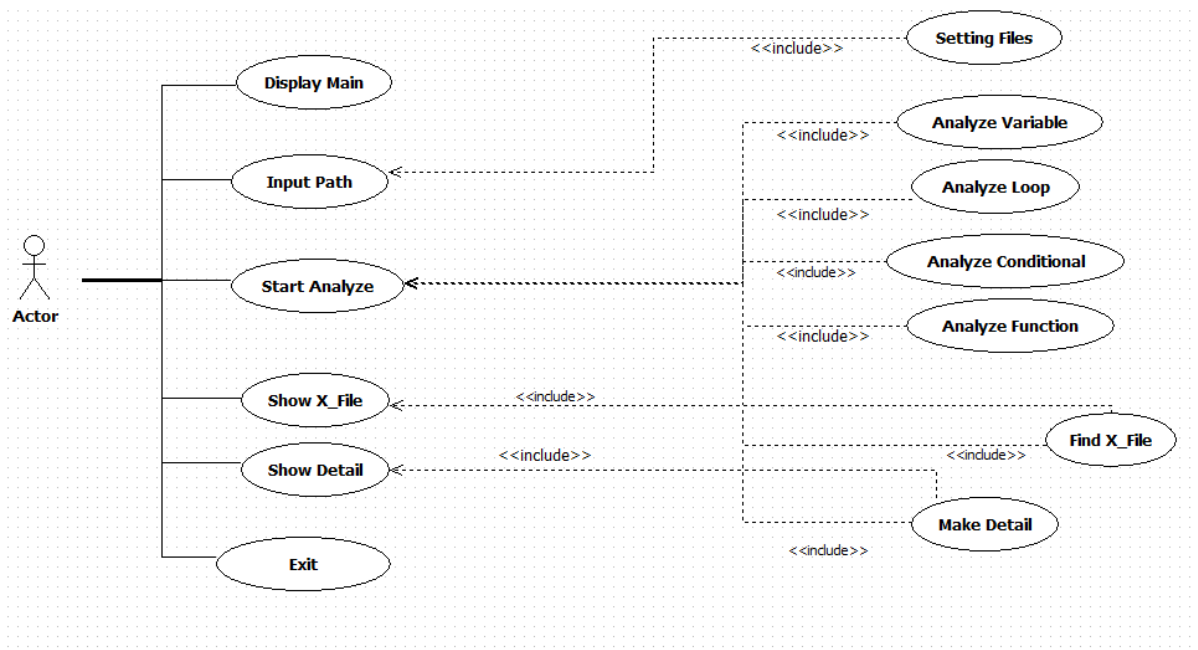
**B. Event based****4. Allocate system functions into Related Use-Case**

Ref. #	Function	Use-Case
R 1.1	Display Main	Display Main
R 1.2	Input Path	Input Path
R 1.2.1	Setting Files	Setting Files
R 2.1	Start Analyze Code	Start Analyze
R 2.1.1	Analyze Variable	Analyze Variable
R 2.1.2	Analyze Loop	Analyze Loop
R 2.1.3	Analyze Conditional	Analyze Conditional
R 2.1.4	Analyze Function	Analyze Function
R 2.1.5	Make Detail	Make Detail
R 2.1.6	Find X_File	Find X_File
R 3.1	Show X_File	Show X_File
R 3.2	Show Detail	Show Detail
R 4.1	Exit	Exit

### 5. Categorize Use-Case

Use-Case	Category
Display Main	Primary
Input Path	Primary
Setting Files	Primary
Start Analyze	Primary
Analyze Variable	Primary
Analyze Loop	Primary
Analyze Conditional	Primary
Analyze Function	Primary
Make Detail	Primary
Find X_File	Primary
Show X_File	Primary
Show Detail	Primary
Exit	Primary

### 6. Draw a Use-Case diagram



## 7. Describe Use-Case

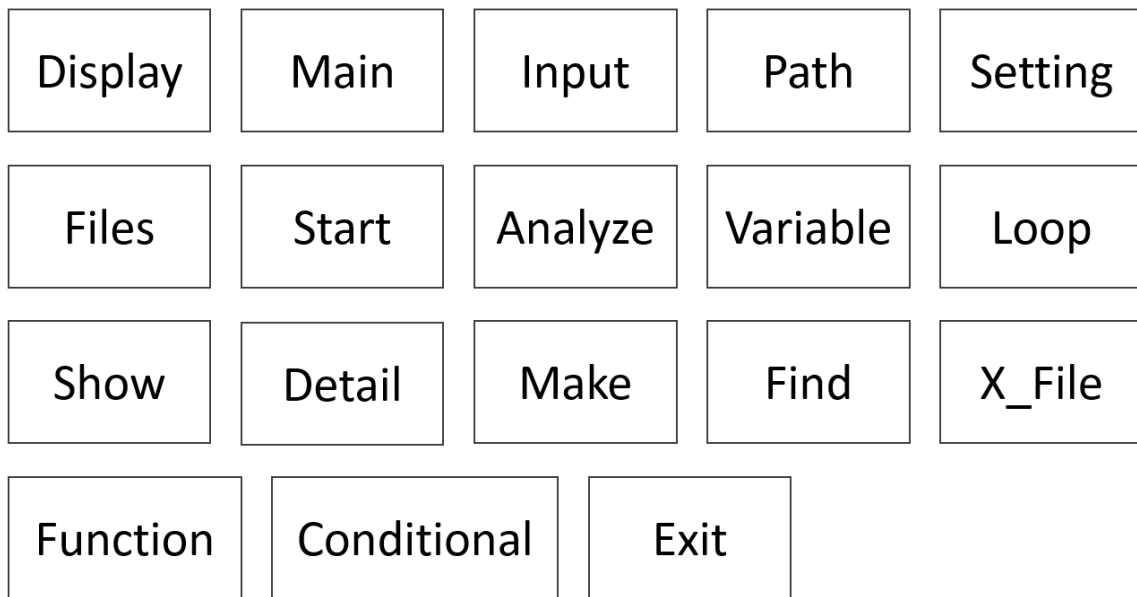
<b>Use Case Name</b>	<b>Display Main</b>
Actor	Actor
Description	프로그램 실행 시 UI를 포함한 초기화면
<b>Use Case Name</b>	<b>Input Path</b>
Actor	Actor
Description	검사 할 파일들이 저장되어 있는 폴더의 경로를 입력한다
<b>Use Case Name</b>	<b>Setting Files</b>
Actor	System
Description	분석에 앞서 파일들을 정리한다
<b>Use Case Name</b>	<b>Start Analyze</b>
Actor	Actor
Description	분석을 시작한다
<b>Use Case Name</b>	<b>Analyze Variable</b>
Actor	System
Description	변수에 대한 검사를 한다
<b>Use Case Name</b>	<b>Analyze Loop</b>
Actor	System
Description	반복문을 검사한다
<b>Use Case Name</b>	<b>Analyze Conditional</b>
Actor	System
Description	조건문을 검사한다
<b>Use Case Name</b>	<b>Analyze Function</b>
Actor	System
Description	함수에 대한 검사를 한다
<b>Use Case Name</b>	<b>Make Detail</b>
Actor	System
Description	가중치를 적용하여 최종 유사도 점수를 계산한 후, 검사의 세부 내용을 .txt 파일에 저장한다.
<b>Use Case Name</b>	<b>Find X_File</b>
Actor	System
Description	최종 유사도 점수들을 바탕으로 X_File 을 찾는다.
<b>Use Case Name</b>	<b>Show X_File</b>
Actor	Actor

Description	X_File 파일명을 보여준다
<b>Use Case Name</b>	<b>Show Detail</b>
Actor	Actor
Description	분석 결과 세부내용 파일(.txt)을 보여준다
<b>Use Case Name</b>	<b>Exit</b>
Actor	Actor
Description	분석 프로그램을 종료한다.

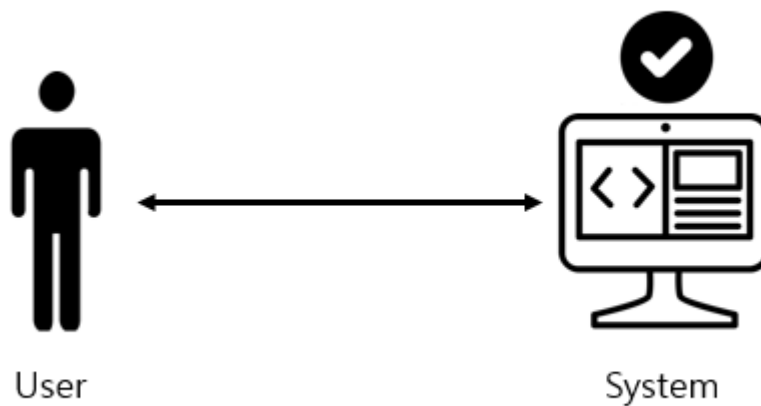
## 8. Rank Use-Case

Use-Case	Rank
Display Main	High
Input Path	High
Setting Files	High
Start Analyze	High
Analyze Variable	High
Analyze Loop	High
Analyze Conditional	High
Analyze Function	High
Make Detail	High
Find X_File	High
Show X_File	High
Show Detail	High
Exit	High

### Activity1007. Define Business Concept Model



### Activity1008. Define Draft System Architecture



## Activity1009. Define System Test Case

Identifier	Feature
CK.STC.110	프로그램의 시작으로 <b>Input 버튼, Exit 버튼만 활성화</b> 되어 있는지 확인한다.
CK.STC.120	<b>존재하지 않는</b> 경로를 입력하고 Input 버튼을 누른 경우, 알림창이 뜨는지 확인한다.
CK.STC.121	<b>존재하는</b> 경로를 입력하고 Input 버튼을 누른 경우, 알림창이 뜨는지 확인한다.
CK.STC.122	<b>.c 파일 외의 다른 파일이 들어있는</b> 경로를 입력하고 Input 버튼을 누른 경우, 알림창이 뜨는지 확인한다.
CK.STC.123	<b>.c 파일만 존재하는</b> 경로를 입력하고 Input 버튼을 누른 경우, 알림창이 뜨는지 확인한다.
CK.STC.124	경로를 입력하고 Input 버튼을 누르고, 경로 탐색 성공 알림창이 뜬 후에 <b>Start 버튼이 활성화</b> 되는지 확인한다.
CK.STC.125	잘못된 경로를 입력하고 Input 버튼을 누르고, 경로 탐색 실패 알림창이 뜬 후에 <b>Start 버튼이 비활성화</b> 상태인지 확인한다.
CK.STC.210	경로 입력을 성공하고 Start 버튼을 누른 경우, 분석 진행 중 알림창이 뜨는지 확인한다.
CK.STC.220	경로 입력을 성공하고 Start 버튼을 누르고, 분석 진행 중 알림창이 뜬 상태에서 Show X_File 과 Show Detail 버튼이 눌리는지 확인한다.
CK.STC.230	경로 입력을 성공하고 Start 버튼을 누르고, 분석 완료 알림창이 뜬 뒤 확인 버튼을 누르면 <b>Start 버튼은 비활성화 되고 Show X_File 과 Show Detail 버튼이 활성화</b> 되는지 확인한다.
CK.STC.310	경로 입력을 성공하고 Start 버튼을 누르고, 분석 완료 알림창이 뜬 뒤에 Show X_File 버튼을 누르면 X_File의 파일명이 보이는지 확인한다.
CK.STC.311	경로 입력을 성공하고 Start 버튼을 누르고, 분석 완료 알림창이 뜬 뒤에 Show X_File 버튼을 누르고 확인 버튼을 누르면 <b>Show X_File 버튼이 비활성화</b> 되는지 확인한다.
CK.STC.312	내용이 동일한 2개의 소스 코드와 내용이 다른 1개의 소스 코드가 존재하는 경로를 입력하고 Input 버튼을 누른 뒤에, Start 버튼을 누르고 분석 완료 알림창이 뜬 뒤에 Show X_File 버튼을 누른 경우, 동일한 2개의 소스 코드 중 하나의 이름이 나오는 지 확인한다.
CK.STC.320	경로 입력을 성공하고 Start 버튼을 누르고, 분석 완료 알림창이 뜬 뒤에 Show Detail 버튼을 누르면 검사의 세부 내용이 담긴 .txt 파일의 내용이 보이는지 확인한다.
CK.STC.321	경로 입력을 성공하고 Start 버튼을 누르고, 분석 완료 알림창이 뜬 뒤에 Show Detail 버튼을 눌러서 생긴 창을 닫으면 <b>Show Detail 버튼이 비활성화</b> 되는지 확인한다.

CK.STC.322	내용이 동일한 5 개의 소스 코드가 존재하는 경로를 입력하고 Input 버튼을 누른 뒤에, Start 버튼을 누르고 분석 완료 알림창이 뜬 뒤에 Show Detail 버튼을 누른 경우, 검사 세부 내용이 모두 동일한지 확인한다.
CK.STC.410	Exit 버튼을 눌렀을 때 확인창이 뜨는지 확인한다.
CK.STC.411	Exit 버튼을 누르고 확인창이 뜬 뒤에, 확인 버튼을 누르면 프로그램이 종료 되는지 확인한다.
CK.STC.412	Exit 버튼을 누르고 확인창이 뜬 뒤에, 취소 버튼을 누르면 프로그램이 종료 되지 않는지 확인한다.

## Activity1010. Refine Plan

### 1. Project Scope

- 다른 사람의 코드를 가져다 쓰는 사람들이 많이 있다. 컴퓨터공학과 학생으로써 코드를 직접 구현하지 않고 다른 사람의 코드를 그대로 가져다 쓰는 것은 실력 향상에 문제가 되기 때문에 하지 않는 것이 좋은 행동이기에 그런 행동들을 하지 못하도록 Clone Checker 를 만들어서, 검사를 하도록 한다.

### 2. Project Objectives

- 실제 컴퓨터공학과 학생들의 과제 제출 시에 자신이 직접 구현하지 않고 다른 학생의 코드를 복사하여 낼 때 사용하는 방법들을 조사한 뒤, 그것을 기준으로 세워서 두 프로그램 코드의 유사도를 검사하는 프로그램이다. 각 기준 마다 가중치를 두어 프로그램을 실행하면 유사도를 비율(%)로 사용자에게 알려주는 것을 목표로 한다.

### 3. Functional Requirements

- Display Main
- Input Path
- Setting Files
- Start Analyze Code



- Analyze Variable
- Analyze Loop
- Analyze Conditional
- Analyze Function
- Make Detail
- Find X\_File
- Show X\_File
- Show Detail
- Exit

#### **4. Performance Requirements**

- A. 50 개의 파일을 1 분 이내로 검사하고 원본으로 추정되는 파일을 찾는다

#### **5. Operating Environment**

OS : Windows 7

IDE : Eclipse Mars2 Release(4.5.2)

개발 언어 : JAVA 1.8.0\_77

#### **6. Resources**

- Human Resource : 3 명
- Project Duration : 16 주
- Cost : 100 만원

## 7. Scheduling

Stage	Phase(00X0)/Activity(000X)	Schedule(Week)													
		1	2	3	4	5	6	7	8	9	10				
1000. Plan & Elaboration	1001. Define Draft Plan	█													
	1002. Create Preliminary Investigation Report	█	█												
	1003. Define Requirements		█												
	1004. Record Terms in Glossary			█											
	1005. Implement Prototype				█										
	1006. Define Business Use Case					█									
	1007. Define Business Concept Model						█								
	1008. Define Draft System Architecture							█							
	1009. Define System Test Case								█						
	1010. Refine Plan									█					
2000. Build	2010. Revise Plan														
	2020. Synchronize Artifacts														
	2030. Analyze														
	2031. Define Essential Use Case														
	2032. Refine Use Case Diagram														
	2033. Refine Conceptual Model														
	2034. Refine Glossary														
	2035. Define System Sequence Diagram														
	2036. Define Operation														
	2037. Define State Diagrams														
	2040. Design														
	2041. Define Real Use Case														
	2042. Define Reports UI and Story boards														
	2043. Refine System Architecture														
	2044. Define Interaction Diagrams														
	2045. Define Design Class Diagrams														
	2046. Define Database Schema														
	2050. Construct														
	2051. Implement Class & Interface Definition														
	2052. Implement Methods														
	2053. Implement Windows														
	2054. Implement Reports														
	2055. Implement DB Schema														
	2056. Write Test Code														
	2060. Test														
	2061. Unit Testing														
2062. Integration Testing															
2063. System Testing															
2064. Performance Testing															
2065. Acceptance Testing															
2066. Documentation Testing															