OSP Stage 1000

<Plan & Elaboration>

그놈! Clone Checker

Project Team

T4

Date

2016-03-31

Team Information

201411258 강태준
201411265 김서우
201411321 홍유리

Contents

Activity1001. Define Draft Plan

1. Motivation

여태껏 2 년동안 학기 중에 크고 작은 실습 과제를 하면서 서로의 코드를 복사하여 제출하는 학생들이 많이 있었다. 물론 우리 팀원들도 1 학년 때는 그 방법을 많이 사용하였었다. 하지만 2 학년 2 학기가 끝나고 3 학년 1 학기를 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 바램에서 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 컴퓨터공학과의 학생들이 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 백분율(%)로 사용자에게 알려주는 것을 목표로 한다.

4. Functional Requirements

- Display Main
- Input Path

- Load Files
- Delete Annotation
- Change Capital
- Start Analyze
- Analyze Change Name
- Analyze for To while
- Analyze while To for
- Analyze if To switch
- Analyze switch To if
- Analyze Separate Functions
- Analyze Combine Functions
- Calculate Similarity
- Save Detail
- Find X File
- Show X File
- Show Detail
- Exit

5. Non-Functional Requirements

A. 단계별 유사도 검사 속도가 빨라야 한다.

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)

7. Other Information

A. Future Version : 코드의 변화 과정을 실시간으로 사용자가 볼 수 있도록 UI를 업데이트한다.

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 : 50 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

2016 년 6 월까지 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.
Load Files	파일을 불러온다.
Delete Annotation	파일의 모든 주석을 삭제한다.
Change Capital	파일의 모든 문자를 대소문자 구분 없이 소문자로 통합한다.
Start Analyze	분석을 시작한다.
Analyze Change Name	변수와 함수 이름을 바꾸었는지 검사한다.
Analyze for To while	for 문을 while 문으로 바꾸었는지 검사한다.
Analyze while To for	while 문을 for 문으로 바꾸었는지 검사한다.
Analyze if To switch	if 문을 switch 문으로 바꾸었는지 검사한다.
Analyze switch To if	switch 문을 if 문으로 바꾸었는지 검사한다.
Analyze Separate Functions	함수를 고의로 나누었는지 검사한다.
Analyze Combine Functions	함수를 고의로 합쳤는지 검사한다.
Calculate Similarity	미리 정해 놓은 단계별 가중치에 따라 최종 유사도를 계산한다
Save Detail	분석한 결과들을 저장한다.
Find X File	결과를 토대로 원본파일을 찾는다
Show X File	결과를 보여준다..
Show Detail	자세한 분석 결과를 보여준다.
Exit	유사도 검사 프로그램을 종료한다.

Ref. #	Function	Category
R 1.1	Display Main	Evident
R 1.2	Input Path	Evident
R 1.3	Load Files	Hidden
R 1.3.1	Delete Annotation	Hidden
R 1.3.2	Change Capital	Hidden
R 2.1	Start Analyze	Evident
R 2.2	Analyze Change Name	Hidden
R 2.2.1	Analyze for To while	Hidden
R 2.2.2	Analyze while To for	Hidden
R 2.2.3	Analyze if To switch	Hidden
R 2.2.4	Analyze switch To if	Hidden
R 2.2.5	Analyze Separate Functions	Hidden
R 2.2.6	Analyze Combine Functions	Hidden
R 3.1	Calculate Similarity	Hidden
R 3.1.1	Save Detail	Hidden
R 3.1.2	Find X File	Hidden
R 4.1	Show X File	Evident
R 4.2	Show Detail	Evident
R 5.1	Exit	Evident

2. Operating Environments

OS : Windows

IDE : Eclipse

3. Development Environments

OS : Windows

개발 언어 : JAVA

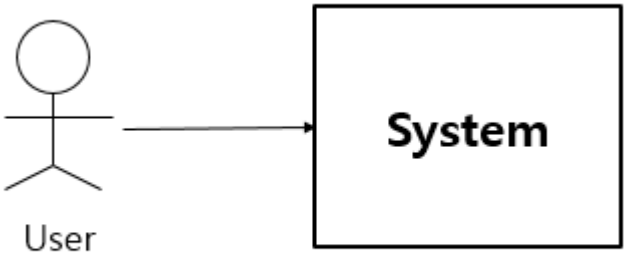
.

Activity1004. Record Terms in Glossary

Term	Description	Remarks
Display Main	프로그램 시작 후 초기화면	
Path	검사하려는 파일들이 있는 폴더의 경로	
Code	복제된지 확인하려는 코드	
Clone	복제품	
X File	원본에 가장 가까운 것 같은 코드	
Annotation	주석(메시지나 설명)	
Capital	대문자	
Analyze	분석하다	
For	반복문의 한 종류	
While	반복문의 한 종류	
if	조건문의 한 종류	
Switch	조건문의 한 종류	
Separate Functions	나눠진 함수	
Combine Functions	합친 함수	
Calculate Similarity	계산된 최종 유사도	
Save	.txt 파일로 결과를 저장	
Show X File	X File 을 보여준다.	
Show Detail	분석결과를 보여준다.	
Exit	프로그램 종료	
그놈 파일	여러 파일들 중에서 가장 중심이 되는 파일	

Activity1006. Define Business Use Case

1. Define System Boundary



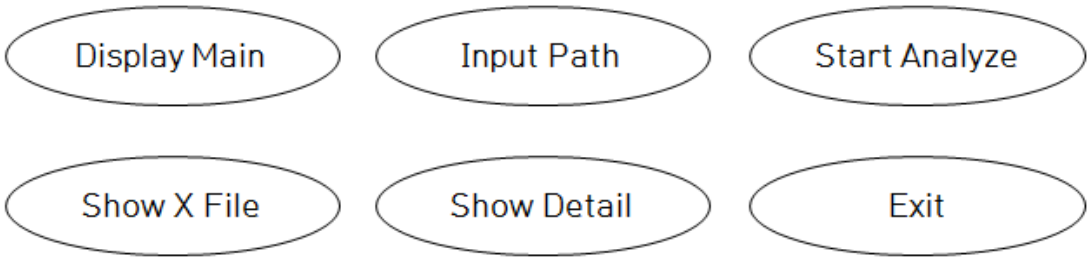
2. Identify and Describe Actors

A. User

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

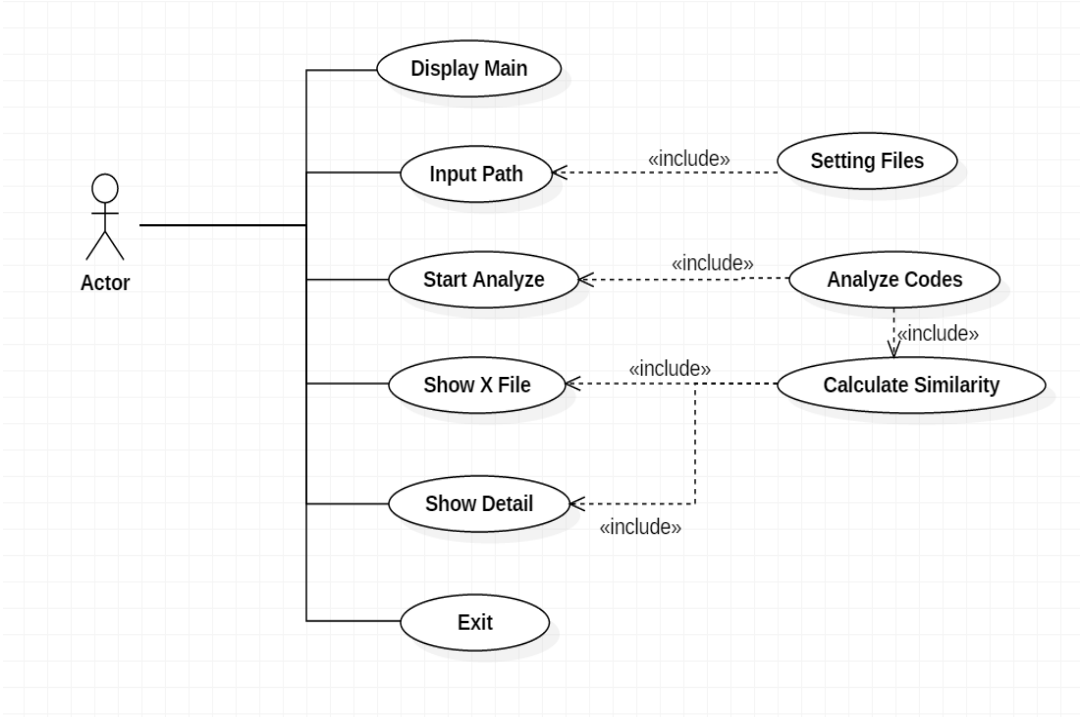
Ref. #	Function	Use-Case
R 1.1	Display Main	Display Main
R 1.2	Input Path	Input Path
R 1.3	Load Files	Setting Files
R 1.3.1	Delete Annotation	Setting Files
R 1.3.2	Change Capital	Setting Files
R 2.1	Start Analyze	Start Analyze
R 2.2	Analyze Change Name	Analyze Codes

R 2.2.1	Analyze for To while	Analyze Codes
R 2.2.2	Analyze while To for	Analyze Codes
R 2.2.3	Analyze if To switch	Analyze Codes
R 2.2.4	Analyze switch To if	Analyze Codes
R 2.2.5	Analyze Separate Functions	Analyze Codes
R 2.2.6	Analyze Combine Functions	Analyze Codes
R 3.1	Calculate Similarity	Calculate Similarity
R 3.1.1	Save Detail	Calculate Similarity
R 3.1.2	Find X File	Calculate Similarity
R 4.1	Show X File	Show X File
R 4.2	Show Detail	Show Detail
R 5.1	Exit	Exit

5. Categorize Use-Case

Use-Case	Category
Display Main	Primary
Input Path	Primary
Setting Files	Primary
Start Analyze	Primary
Analyze Codes	Primary
Show X File	Primary
Calculate Similarity	Primary
Show Detail	Primary
Exit	Primary

6. Draw a Use-Case diagram



7. Describe Use-Case

Use Case Name	Display Main
Actor	User
Description	프로그램 실행 시 UI를 포함한 초기화면
Use Case Name	**Input Path**
Actor	User
Description	검사 할 파일들이 저장되어 있는 폴더의 경로를 입력한다.
Use Case Name	**Setting Files**
Actor	System
Description	분석에 앞서 파일들을 정리한다.
Use Case Name	**Start Analyze**
Actor	User
Description	분석을 시작한다.
Use Case Name	**Analyze Codes**
Actor	System
Description	코드를 분석한다.
Use Case Name	**Calculate Similarity**
Actor	System

Description	분석한 결과를 토대로 유사도를 계산하여 저장하고, 원본파일에 유사한 것을 찾는다.
Use Case Name	**Show X File**
Actor	User
Description	그놈 파일 파일명을 보여준다.
Use Case Name	**Show Detail**
Actor	User
Description	분석내용을 보여준다.
Use Case Name	**Exit**
Actor	User
Description	분석 프로그램을 종료한다.

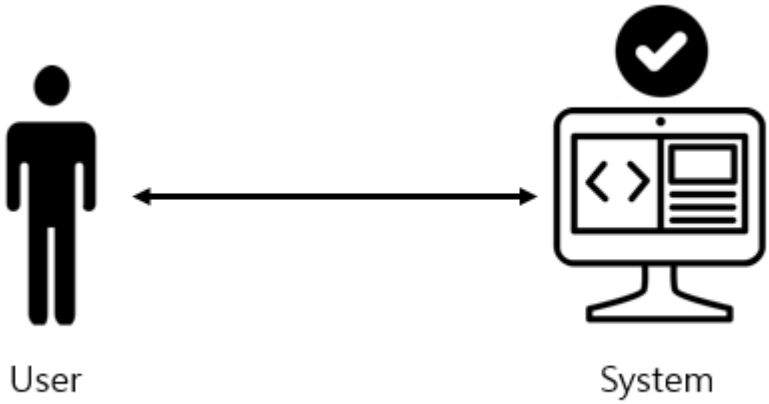
8. Rank Use-Case

Use-Case	Rank
Display Main	High
Input Path	High
Setting Files	High
Start Analyze	High
Analyze Codes	High
Show X File	High
Calculate Similarity	High
Show Detail	High
Exit	High

Activity1007. Define Business Concept Model

Start	Display	Main	Input	Path
Show	Detail	File	Setting	Delete
Annotation	Change	Capital	Analyze	Name
If	Switch	For	While	Separate
Combine	Function	Calculate	Similarity	Code
Save	Result	Find	X File	Exit

Activity1008. Define Draft System Architecture



Activity1009. Define System Test Case

Identifier	Feature
CK.STC.100	Main 화면이 제대로 실행이 되었는지 검사한다.
CK.STC.101	사용자가 입력한 경로가 정확하게 입력되었는지 검사한다.
CK.STC.102	사용자가 입력한 폴더의 파일들이 잘 불러졌는지 검사한다.
CK.STC.200	파일 내의 모든 주석이 지워졌는지 검사한다.
CK.STC.201	파일 내의 모든 문자가 소문자로 바뀌었는지 검사한다.
CK.STC.211	변수 명을 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.212	함수 명을 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.213	if 문을 switch 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.214	switch 문을 if 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.215	for 문을 while 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.216	while 문을 for 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.217	함수를 고의적으로 나눈 코드를 잘 찾아 냈는지 검사한다.
CK.STC.218	함수를 고의적으로 합친 코드를 잘 찾아 냈는지 검사한다.
CK.STC.300	단계별 가중치를 적용한 유사도 백분율이 알맞게 계산이 되었는지 검사한다.
CK.STC.301	유사도 검사가 끝났을 때 마다 결과를 작성한 텍스트 파일이 맞게 생성되었는지 검사한다.
CK.STC.302	유사도 검사를 통하여 코드들 중 가장 중심이 될 것이다 예상되는 코드를 알맞게 정햇는지 검사한다.
CK.STC.303	유사도 검사를 통하여 코드들 중 가장 중심이 될 것이다 예상되는 파일명을 제대로 보여주는지 검사한다
CK.STC.400	유사도 검사가 끝난 뒤에 사용자가 원한대로 종료가 되는 지를 검사한다.

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 것 이 좋은 행동이기에 그런 행동들을 하지 못하도록 Clone Checker 를 만들어서, 검사를 하도록 한다.

2. Project Objectives

- 실제 컴퓨터공학과의 학생들이 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 백분율(%)로 사용자에게 알려주는 것을 목표로 한다.

3. Functional Requirements

- Display Main
- Input Path
- Load Files
- Delete Annotation
- Change Capital
- Start Analyze
- Analyze Change Name
- Analyze for To while
- Analyze while To for
- Analyze if To switch

- Analyze switch To if
- Analyze Separate Functions
- Analyze Combine Functions
- Calculate Similarity
- Save Detail
- Find X File
- Show X File
- Show Detail
- Exit

4. Performance Requirements

- 단계별 유사도 검사 속도가 빨라야 한다.

5. Operating Environment

- OS : Window
- IDE : Eclipse

6. Other Requirements

- 코드의 변화 과정을 실시간으로 사용자가 볼 수 있도록 UI 를 업데이트 한다

7. Resources

- Human Resource : 3 명

- Project Duration : 16 주

- Cost : 100 만원

8. 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 Reportsm UI and Story boards																				
	2043. Refine System Architecture																				
	2044. Define Interation 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																				