

# OOPT Stage 1000

## <Plan and Elaboration>

Software Modeling & Analysis

소프트웨어 모델링 및 분석

보고서 Version. 3

### Team. T1

201111388 조연호

201211374 이창오

201211379 장종훈

201314196 양동혁

# Stage 1000. Plan and Elaboration

1. Activity 1001. Define Draft Plan .....	3
2. Activity 1002. Create Preliminary Investigation Report.....	4
3. Activity 1003. Define Requirements.....	5
4. Activity 1004. Record Terms in Glossary.....	8
5. Activity 1005. Implement Prototype .....	8
6. Activity 1006. Define Draft System Architecture.....	9
7. Activity 1007. Define Business Use Case.....	9
8. Activity 1008. Define Business Concept Model .....	17
9. Activity 1009. Define System Test Case .....	17
10. Activity 1010. Refine Plan .....	19

## Activity 1001. Define Draft Plan

### 1. Motivation

시중의 Clone Checker 프로그램들은 대부분 소스 코드 파일 두 개를 비교한다. 반대로 다수의 소스 코드 파일을 한 번에 검사하는 Clone Checker 프로그램은 그 결과를 보여주는 데 있어서 단순한 결과만을 출력할 뿐 각각 소스 코드 파일의 인과 관계를 보여주는 것이 미흡하다. 그러나 정작 중요한 것은 어떤 소스 코드 파일을 기준으로 그것과 비슷한 소스 코드 파일들을 찾아내어 복제와 연관된 인과 관계를 찾아내는 것이다. 이러한 인과 관계를 직관적이며 편하게 볼 수 있는 프로그램이 필요하다고 판단하였다.

### 2. Project Objectives

하나의 소스 코드 파일을 기준으로 설정하고, 다른 소스 코드 파일과의 일치율을 알기 쉽게 보여주는 Clone Checker 프로그램을 개발하여 비교 정확도를 높이는 것을 목표로 한다.

### 3. Functional Requirements

- A. 초기화
- B. 폴더 지정
- C. 분석 시작
- D. 결과 출력
- E. 비교 기준 소스 코드 파일 변경

### 4. Non-Function Requirements

- A. 설명이 필요 없을 정도의 조작하기 쉬운 인터페이스
- B. 한 눈에 이해 가능한 결과 출력
- C. 다른 언어의 소스 파일도 비교 가능하도록 하는 확장성

### 5. Resource Estimation

- A. Human Resource : 4명
- B. Project Duration : 3개월(12주)
- C. Human Efforts(Man-Month) : 12
- D. Cost : 식대 1,200,000원 (5,000원/일 × 4명 × 5일/주 × 12주)

## 6. Other Information

### A. Future Version

다양한 시각에서 엄격한 비교 기준들을 추가하여 두 소스 코드 파일 내에서 프로그램 구현에 사용한 동일 알고리즘을 찾아낼 수 있는 Clone Checker 프로그램으로 확장한다.

## Activity 1002. Create Preliminary Investigation Report

### 1. Alternative Solutions

- A. 프로그램 외주 제작
- B. 과제 대행 서비스
- C. 현재 존재하는 Clone Checker 프로그램

### 2. Project Justification(Business Demands)

- A. Cost : 식대 외의 다른 비용이 필요하지 않아 다른 대안보다 훨씬 저렴하다.
- B. Duration : 3개월(12주)
- C. Risk : UML 및 JAVA 숙련도 부족, 개인적인 사정 등
- D. Effect

전체 소스 코드 파일들의 비교가 아닌 기준 소스 코드 파일 하나에 초점을 맞추어 진행하여 원본 파일을 제공한 학생과 상습적으로 복제를 하는 학생을 쉽게 찾아낼 수 있다.

### 3. Risk Management

Risk	Probability	Significance	Weight
UML 숙련도 부족	4	5	20
JAVA 숙련도 부족	5	5	25
Clone Checker 원리 이해도 부족	4	4	16
아르바이트	3	4	12
동아리 및 학생회 활동	3	4	12
팀원 간 의견 충돌	5	4	20
연애 사업	1	8	8

4. Risk Reduction Plan

Risk	Way of Reduction
UML 숙련도 부족	적극적인 수업 참여와 관련 도서 및 인터넷을 통하여 학습한다
JAVA 숙련도 부족	적극적인 수업 참여와 관련 도서 및 인터넷을 통하여 학습한다
Clone Checker 원리 이해도 부족	관련 도서 및 인터넷을 통하여 학습한다.
아르바이트	각자 조정할 수 있는 스케줄은 조정한다.
동아리 및 학생회 활동	각자 조정할 수 있는 스케줄은 조정한다.
팀원 간 의견 충돌	팀장이 강력한 리더십을 발휘한다.
연애 사업	연애 문제로 인한 감정 기복이 생기지 않도록 연애 사업을 하는 팀원을 적극적으로 도와준다.

5. Market Analysis

창의적인 Clone Checker 프로그램으로써 아직까지 비슷한 대체제가 없어 시장성이 높다.

6. Other Managerial Issues

- A. 2016년 6월 안에 프로젝트가 종료되어야 한다.
- B. 다음 학기부터 곧바로 이용할 수 있도록 완벽성을 추구해야 한다.

Activity 1003. Define Requirements

1. Functional Requirements

Ref	Function	Description
R.1	Display Main	폴더를 지정할 수 있는 버튼과 '시작하기' 버튼이 제공되는 메인 화면을 출력한다.
R.1.1	Init	프로그램을 초기화한다.
R.1.2	Select Folder	비교할 소스 코드 파일들이 들어있는 폴더를 지정한다.
R.2	Display Result	기준 소스 코드 파일을 기준으로 태그 클라우드형 비교 결과 화면을 출력한다.
R.2.1	Analyze File	소스 코드 파일을 분석하여 일치율 계산에 필요한 정보로 가공한다.
R.2.1.1	Analyze Line	소스 코드 파일의 라인 수를 분석한다.
R.2.1.2	Make AST	소스 코드 파일의 AST(Abstract Syntax Tree)를 생성한다.
R.2.1.2.1	Analyze Function	생성된 AST(Abstract Syntax Tree)에서 함수의 개수와 이름을 분석한다.

**OOPT Stage 1000 <Plan and Elaboration>**

R.2.1.2.2	Analyze Variable	생성된 AST(Abstract Syntax Tree)에서 변수의 개수와 이름을 분석한다.
R.2.1.2.3	Analyze Preprocessor	생성된 AST(Abstract Syntax Tree)에서 전처리기의 개수와 이름을 분석한다.
R.2.1.2.4	Analyze Annotation	생성된 AST(Abstract Syntax Tree)에서 주석의 개수를 분석한다.
R.2.2	Change Center	비교 기준이 되는 소스 코드 파일을 변경한다.
R.2.2.1	Compare File	분석하여 가공한 정보를 활용(비교)하여 일치율을 계산한다.
R.2.2.1.1	Compare Line	라인 수를 활용(비교)하여 일치율을 계산한다.
R.2.2.1.2	Compare Function	함수의 개수와 이름을 활용(비교)하여 일치율을 계산한다.
R.2.2.1.3	Compare Variable	변수의 개수와 이름을 활용(비교)하여 일치율을 계산한다.
R.2.2.1.4	Compare Preprocessor	전처리기의 개수와 이름을 활용(비교)하여 일치율을 계산한다.
R.2.2.1.5	Compare Annotation	주석의 개수를 활용(비교)하여 일치율을 계산한다.
R.2.3	Display Sync	일치율과 유사 항목을 출력한다.

Ref	Function	Category
R.1	Display Main	Evident
R.1.1	Init	Hidden
R.1.2	Select Folder	Evident
R.2	Display Result	Evident
R.2.1	Analyze File	Hidden
R.2.1.1	Analyze Line	Hidden
R.2.1.2	Make AST	Hidden
R.2.1.2.1	Analyze Function	Hidden
R.2.1.2.2	Analyze Variable	Hidden
R.2.1.2.3	Analyze Preprocessor	Hidden
R.2.1.2.4	Analyze Annotation	Hidden
R.2.2	Change Center	Evident
R.2.2.1	Compare File	Hidden
R.2.2.1.1	Compare Line	Hidden
R.2.2.1.2	Compare Function	Hidden
R.2.2.1.3	Compare Variable	Hidden

## OOPT Stage 1000 <Plan and Elaboration>

R.2.2.1.4	Compare Preprocessor	Hidden
R.2.2.1.5	Compare Annotation	Hidden
R.2.3	Display Sync	Evident

### 2. Performance Requirements

- A. 소스 코드 파일 분석 작업은 5초 이내로 수행되어야 한다.
- B. 기준 소스 코드 파일 변경 시 2초 이내로 비교 결과가 변경되어야 한다.

### 3. Operating Environments

Microsoft Windows 7 이상

### 4. Develop Environments

- A. 운영체제 : Windows 7, 10
- B. CPU : Intel
- C. IDE : Eclipse
- D. 개발 언어 : JAVA
- E. UML 툴 : StarUML

### 5. Interface Requirements

#### A. 메인 화면

##### i. 폴더 지정 버튼

화면 중앙에 위치하며 클릭 시 폴더 선택 화면을 출력한다.

##### ii. 비교 시작 버튼

화면 하단에 위치하며 클릭 시 소스 코드 파일 분석 후 비교 결과 화면을 출력한다.

#### B. 비교 결과 화면

##### i. 기준 소스 코드 파일 이름

화면 중앙에 위치한다.

##### ii. 비교 소스 코드 파일 이름

기준 소스 코드 파일 이름을 중심으로 일치율에 따라 크기가 달리 출력된다.

iii. 일치율과 유사 항목

비교 소스 코드 파일 이름에 마우스 커서를 위치할 경우 일치율과 유사 항목을 출력한다.

iv. 이전 화면 버튼

화면 하단에 위치하며 클릭 시 메인 화면을 출력한다.

C. 그 외 메시지 출력은 대화상자(Dialog) 활용

6. Other Requirements

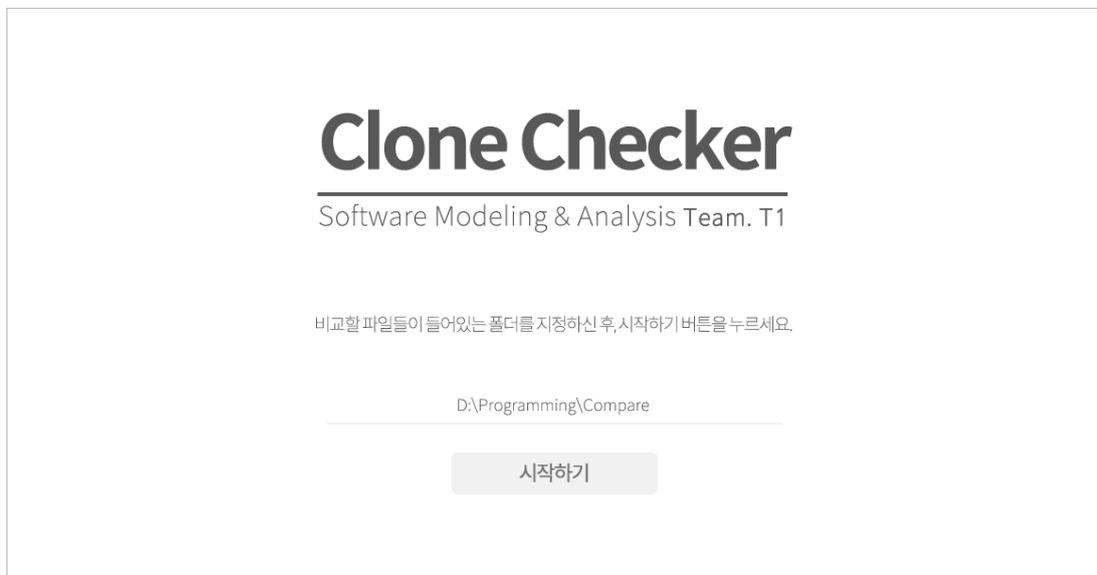
A. 비교 결과(일치율, 유사 항목)를 한 눈에 알아볼 수 있도록 비교 결과 출력 화면이 직관적이어야 한다.

## Activity 1004. Record Terms in Glossary

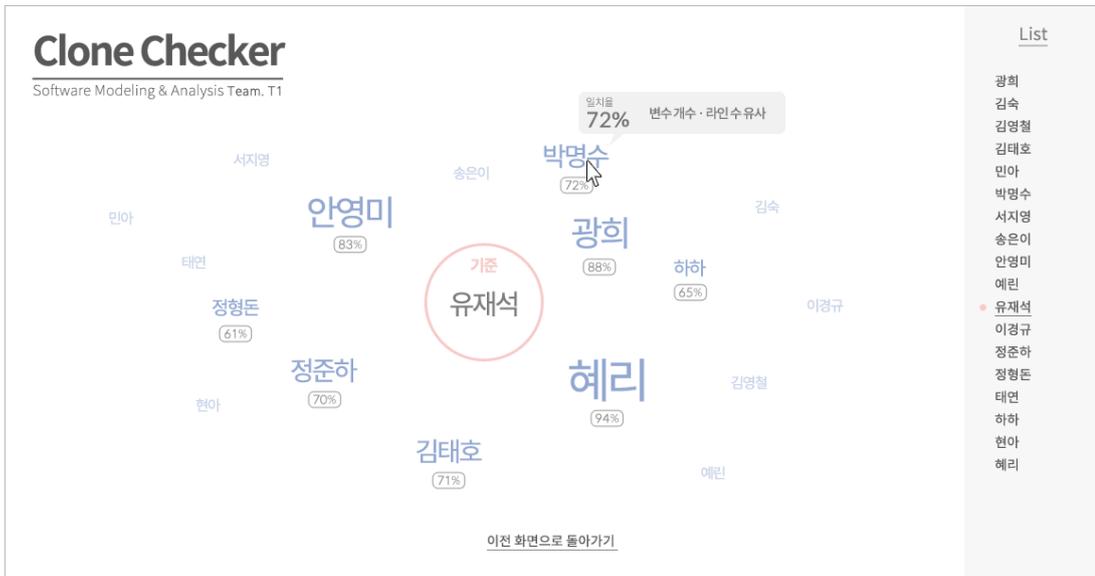
Term	Description	Remarks
Clone	비교 결과 일치율 85% 이상인 파일	

## Activity 1005. Implement Prototype

1. 메인 화면



2. 비교 결과 화면



Activity 1006. Define Draft System Architecture



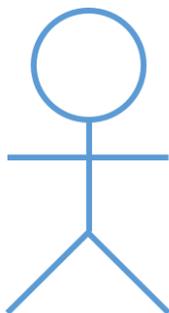
User



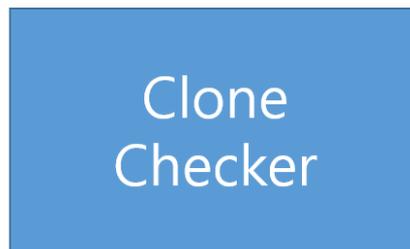
Clone Checker  
(Program)

Activity 1007. Define Business Use Case

1. Define System Boundary



User



Clone Checker  
(Program)

## 2. Identify and Describe Actors

### A. User

User는 다음과 같은 행동을 할 수 있다.

#### i. 메인 화면

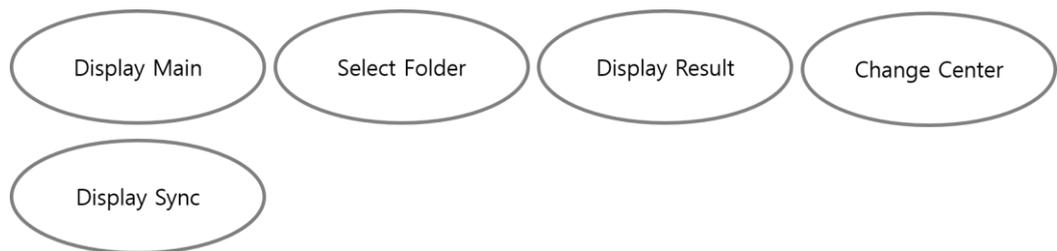
비교하고자 하는 소스 코드 파일들이 들어있는 폴더를 지정하고, '시작하기' 버튼을 클릭하여 유사한 소스 코드 파일이 있는지 비교할 수 있다.

#### ii. 비교 결과 화면

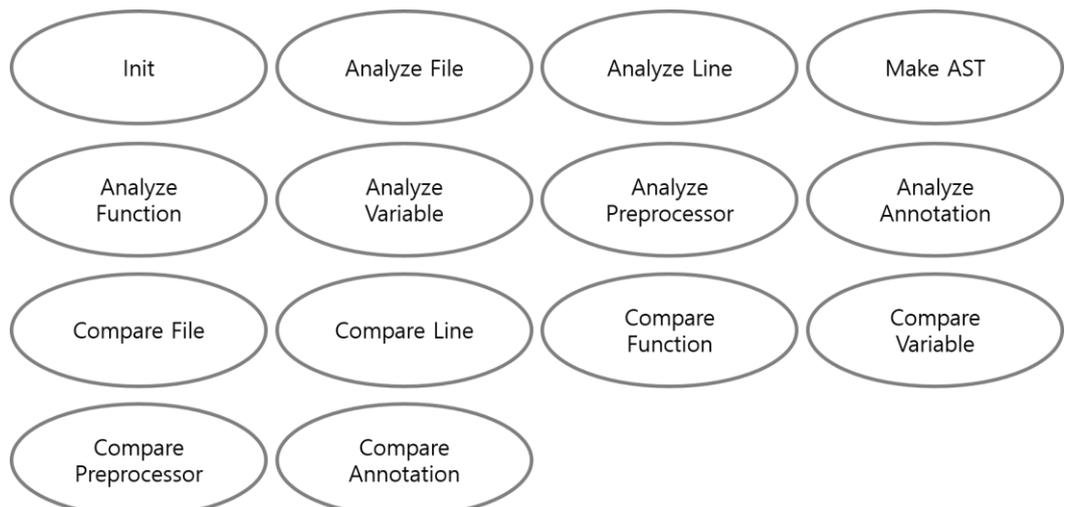
1. 소스 코드 파일 이름에 마우스를 올려 일치율과 유사 항목을 확인한다.
2. 태그 클라우드 또는 오른쪽 리스트에서 소스 코드 파일 이름을 클릭하여 기준을 변경한다.

## 3. Identify Use-Case

### A. Use-Cases by Actor-Based



### B. Use-Cases by Event-Based



**4. Allocate System Functions into Related Use-Cases**

<b>Ref</b>	<b>Function</b>	<b>Use-Case Number &amp; Name</b>
R.1	Display Main	1. Display Main
R.1.1	Init	2. Init
R.1.2	Select Folder	3. Select Folder
R.2	Display Result	4. Display Result
R.2.1	Analyze File	5. Analyze File
R.2.1.1	Analyze Line	6. Analyze Line
R.2.1.2	Make AST	7. Make AST
R.2.1.2.1	Analyze Function	8. Analyze Function
R.2.1.2.2	Analyze Variable	9. Analyze Variable
R.2.1.2.3	Analyze Preprocessor	10. Analyze Preprocessor
R.2.1.2.4	Analyze Annotation	11. Analyze Annotation
R.2.2	Change Center	12. Change Center
R.2.2.1	Compare File	13. Compare File
R.2.2.1.1	Compare Line	14. Compare Line
R.2.2.1.2	Compare Function	15. Compare Function
R.2.2.1.3	Compare Variable	16. Compare Variable
R.2.2.1.4	Compare Preprocessor	17. Compare Preprocessor
R.2.2.1.5	Compare Annotation	18. Compare Annotation
R.2.3	Display Sync	19. Display Sync

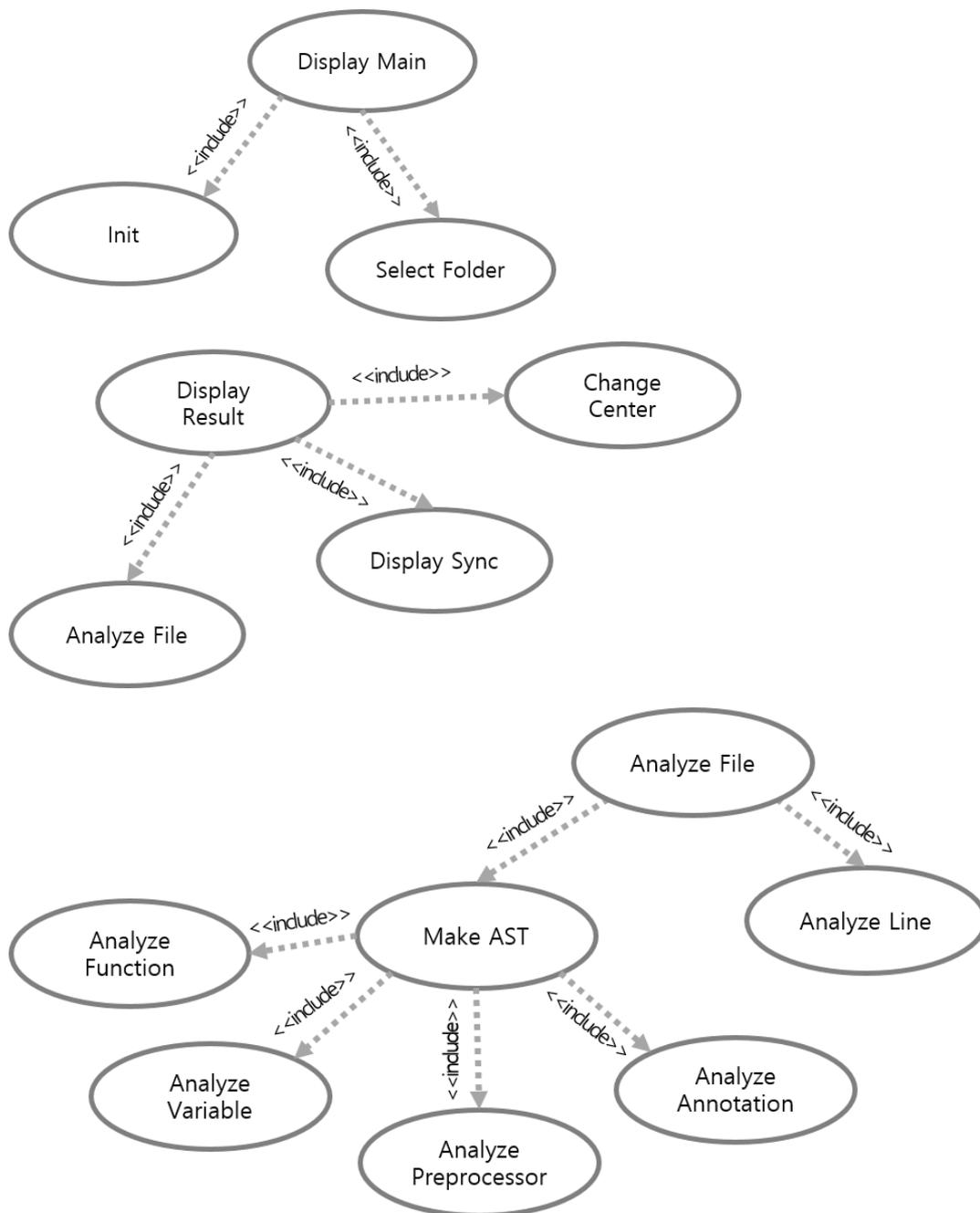
**5. Categorize Use-Cases**

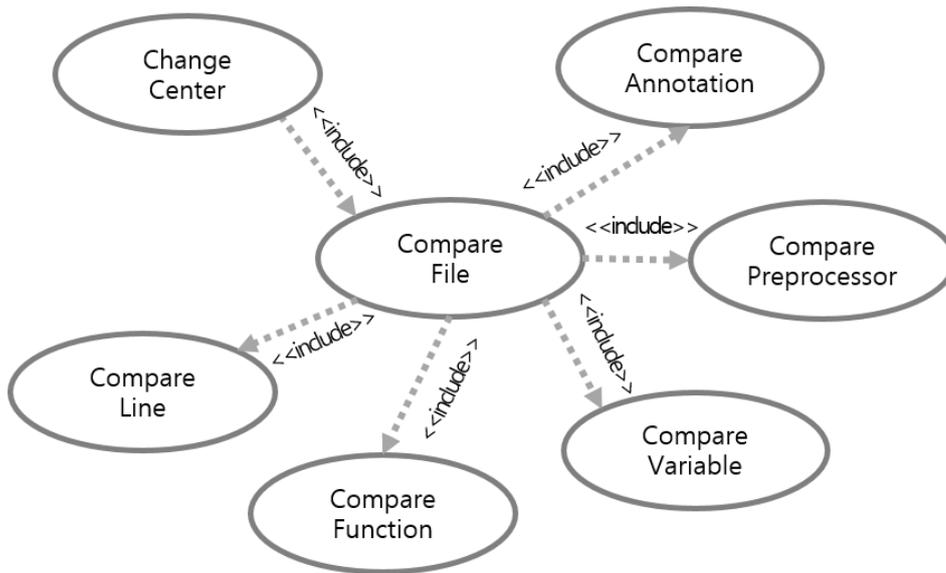
<b>Use-Case Number &amp; Name</b>	<b>Category</b>
1. Display Main	Primary
2. Init	Primary
3. Select Folder	Primary
4. Display Result	Primary
5. Analyze File	Primary
6. Analyze Line	Primary
7. Make AST	Primary
8. Analyze Function	Primary
9. Analyze Variable	Primary
10. Analyze Preprocessor	Primary
11. Analyze Annotation	Primary
12. Change Center	Primary

**OOPT Stage 1000 <Plan and Elaboration>**

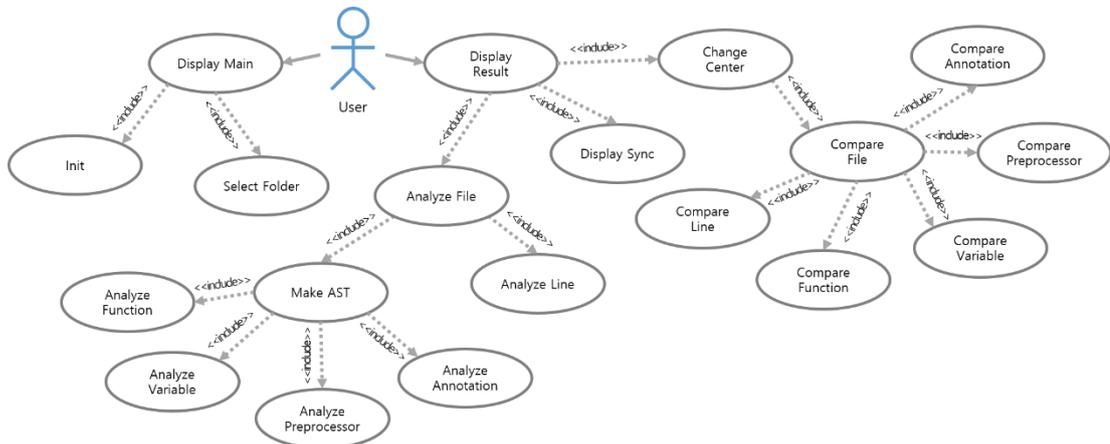
13. Compare File	Primary
14. Compare Line	Primary
15. Compare Function	Primary
16. Compare Variable	Primary
17. Compare Preprocessor	Primary
18. Compare Annotation	Primary
19. Display Sync	Primary

**6. Identify Relationships between Use-Cases**





7. Draw a Use-Case Diagram



8. Describe Use-Cases

Use-Case Name	Description
1. Display Main	폴더를 지정할 수 있는 버튼과 '시작하기' 버튼이 제공된다.
<b>Actor</b>	
User	

Use-Case Name	Description
2. Init	프로그램 실행 시 또는 재실행 시 프로그램의 모든 내용을 초기화한다.
<b>Actor</b>	
None	

**OOPT Stage 1000 <Plan and Elaboration>**

Use-Case Name	Description
3. Select Folder	비교할 소스 코드 파일들이 들어있는 폴더를 지정한다.
Actor	
None	

Use-Case Name	Description
4. Display Result	기준 소스 코드 파일을 기준으로 태그 클라우딩 비교 결과 화면을 출력한다.
Actor	
User	

Use-Case Name	Description
5. Analyze File	소스 코드 파일을 분석하여 일치율 계산에 필요한 정보로 가공한다.
Actor	
None	

Use-Case Name	Description
6. Analyze Line	소스 코드 파일의 라인 수를 분석한다.
Actor	
None	

Use-Case Name	Description
7. Make AST	소스 코드 파일의 AST(Abstract Syntax Tree)를 생성한다.
Actor	
None	

Use-Case Name	Description
8. Analyze Function	생성된 AST(Abstract Syntax Tree)에서 함수의 개수와 이름을 분석한다.
Actor	
None	

Use-Case Name	Description
9. Analyze Variable	생성된 AST(Abstract Syntax Tree)에서 변수의 개수와 이름을 분석한다.
Actor	
None	

**OOPT Stage 1000 <Plan and Elaboration>**

Use-Case Name	Description
10. Analyze Preprocessor	생성된 AST(Abstract Syntax Tree)에서 전처리기의 개수와 이름을 분석한다.
<b>Actor</b>	
None	

Use-Case Name	Description
11. Analyze Annotation	생성된 AST(Abstract Syntax Tree)에서 주석의 개수를 분석한다.
<b>Actor</b>	
None	

Use-Case Name	Description
12. Change Center	비교 기준이 되는 소스 코드 파일을 변경하여 다시 비교한다.
<b>Actor</b>	
None	

Use-Case Name	Description
13. Compare File	분석하여 가공한 정보를 활용(비교)하여 일치율을 계산한다.
<b>Actor</b>	
None	

Use-Case Name	Description
14. Compare Line	라인 수를 활용(비교)하여 일치율을 계산한다.
<b>Actor</b>	
None	

Use-Case Name	Description
15. Compare Function	함수의 개수와 이름을 활용(비교)하여 일치율을 계산한다.
<b>Actor</b>	
None	

Use-Case Name	Description
16. Compare Variable	변수의 개수와 이름을 활용(비교)하여 일치율을 계산한다.
<b>Actor</b>	
None	

**OOPT Stage 1000 <Plan and Elaboration>**

Use-Case Name	Description
17. Compare Preprocessor	전처리기의 개수와 이름을 활용(비교)하여 일치율을 계산한다.
<b>Actor</b>	
None	

Use-Case Name	Description
18. Compare Annotation	주석의 개수를 활용(비교)하여 일치율을 계산한다.
<b>Actor</b>	
None	

Use-Case Name	Description
19. Display Sync	일치율과 유사 항목을 제공한다.
<b>Actor</b>	
None	

**9. Rank Use-Cases**

Use-Case Number & Name	Rank
1. Display Main	High
2. Init	High
3. Select Folder	High
4. Display Result	High
5. Analyze File	High
6. Analyze Line	High
7. Make AST	High
8. Analyze Function	High
9. Analyze Variable	High
10. Analyze Preprocessor	High
11. Analyze Annotation	High
12. Change Center	High
13. Compare File	High
14. Compare Line	High
15. Compare Function	High
16. Compare Variable	High
17. Compare Preprocessor	High
18. Compare Annotation	High
19. Display Sync	High

## Activity 1008. Define Business Concept Model

Interface	File	AST	Compare

## Activity 1009. Define System Test Case

### 1. Functional Requirements Test Case

Ref	Function	Test Case
R.1	Display Main	폴더 지정 버튼과 '시작하기' 버튼이 출력되는지 확인한다.
R.1.1	Init	메인 화면이 출력되는지 확인한다.
R.1.2	Select Folder	폴더 지정 화면이 나타나고, 정상적으로 지정되었는지 확인한다.
R.2	Display Result	태그 클라우드 형으로 비교 결과 출력 화면이 나오는지 확인한다.
R.2.1	Analyze File	비교 결과 출력 화면이 나오는지 확인한다.
R.2.1.1	Analyze Line	소스 코드 파일의 라인 개수만 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.1.2	Make AST	소스 코드 파일의 함수 또는 변수, 전처리기, 주석을 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.1.2.1	Analyze Function	소스 코드 파일의 함수를 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.1.2.2	Analyze Variable	소스 코드 파일의 변수를 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.1.2.3	Analyze Preprocessor	소스 코드 파일의 전처리기를 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.1.2.4	Analyze Annotation	소스 코드 파일의 주석을 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.2	Change Center	기준 소스 코드 파일 이름이 변경되는지 확인한다.

**OOPT Stage 1000 <Plan and Elaboration>**

R.2.2.1	Compare File	비교 소스 코드 파일 이름에 마우스 커서를 올리면 일치율과 유사 항목이 출력되는지 확인한다.
R.2.2.1.1	Compare Line	소스 코드 파일의 라인 개수만 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.2.1.2	Compare Function	소스 코드 파일의 함수를 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.2.1.3	Compare Variable	소스 코드 파일의 변수를 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.2.1.4	Compare Preprocessor	소스 코드 파일의 전처리기를 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.2.1.5	Compare Annotation	소스 코드 파일의 주석을 변경하면서 예상한 결과값과 프로그램 실행 후 얻어지는 결과값을 비교해 본다.
R.2.3	Display Sync	비교 소스 코드 파일 이름에 마우스 커서를 올리면 일치율과 유사 항목이 출력되는지 확인한다.

**2. Non-Functional Requirements Test Case**

Category	Test Case
Interface	<ul style="list-style-type: none"> <li>● 태그 클라우드 출력 화면에서 출력되는 글씨가 겹치지 않는지 확인한다.</li> <li>● 태그 클라우드 출력 화면에서 일치율과 소스 코드 파일 이름의 크기가 비례하는지 확인한다.</li> <li>● 태그 클라우드 출력 화면에서 소스 코드 파일 이름에 마우스 커서를 올릴 경우 일치율과 유사 항목이 정상적으로 출력되는지 확인한다.</li> <li>● 태그 클라우드 출력 화면에서 소스 코드 파일 이름 클릭 시 기준이 변경되는지 확인한다.</li> <li>● 리스트 출력 화면에서 소스 코드 파일 이름 클릭 시 기준이 변경되는지 확인한다.</li> <li>● 리스트 출력 화면에서 소스 코드 파일 이름이 리스트 가로 길이를 넘지 않는지 확인한다.</li> </ul>
Speed	<ul style="list-style-type: none"> <li>● 소스 코드 분석 시간이 5초 이내인지 확인한다.</li> <li>● 일치율과 유사 항목 계산 시간이 2초 이내인지 확인한다.</li> <li>● 비교 결과 화면이 8초 이내인지 확인한다.</li> <li>● 기준 소스 코드 파일 변경 후 2초 이내에 결과가 바뀌는지 확인한다.</li> </ul>

## Activity 1010. Refine Plan

### 1. Project Scope

#### 비교 작업과 결과 출력이 쾌적한 Clone Checker

다수의 소스 코드 파일을 비교하면 시간은 오래 걸리고, 결과 자료의 양도 방대하다. 많은 양의 결과를 확인하는 것은 사람으로 하여금 거부감이 들며 이해하는 데 적지 않은 시간이 걸린다. 따라서 비교 결과가 거부감 없이 명확하고 한 눈에 알아 볼 수 있는 쾌적한 Clone Checker 프로그램을 개발한다.

### 2. Project Objectives

하나의 소스 코드 파일을 기준으로 설정하고, 다른 소스 코드 파일과의 일치율을 알기 쉽게 보여주는 Clone Checker 프로그램을 개발하여 비교 정확도를 높이는 것을 목표로 한다.

### 3. Functional Requirements

- A. 초기화
- B. 폴더 지정
- C. 분석 시작
- D. 결과 출력
- E. 비교 기준 소스 코드 파일 변경

### 4. Performance Requirements

- A. 소스 코드 파일 분석 작업은 5초 이내로 수행되어야 한다.
- B. 기준 소스 코드 파일 변경 시 2초 이내로 비교 결과가 변경되어야 한다.

### 5. Operating Environment

Microsoft Windows 7 이상

### 6. User Interface Requirements

알아보기 쉬운 Interface를 통해 특별한 설명 없이 프로그램을 수행하고, 간결한 결과 화면을 통해 사용자에게 결과를 정확하게 전달한다.

### 7. Other Requirements

다른 프로그래밍 언어의 소스 파일 또는 다른 분야에서도 사용이 가능할 수 있게끔 확장성을 지닐 수 있도록 한다.

**OOPT Stage 1000 <Plan and Elaboration>**

**8. Resources**

- A. Human Resource : 4명
- B. Project Duration : 3개월(12주)
- C. Human Efforts(Man-Month) : 12
- D. Cost : 식대 1,200,000 원 (5,000 원/일 × 4명 × 5 일/주 × 12 주)

**9. Scheduling**

Stage	Phase/Activity	Schedule (Week)																
		1	2	3	4	5	6	7	8	9	10	11	12					
1000 Plan & Elaborate	1001. Define Draft Plan	█																
	1002. Create Preliminary Investigation Report																	
	1003. Define Requirements																	
	1004. Record Terms in Glossary		█															
	1005. Implement Prototype		█															
	1006. Define Draft System Architecture		█															
	1007. Define Business Use Case																	
	1008. Define Business Concept Model																	
	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 Diagrams																
		2033. Refine Conceptual Model																
		2034. Refine Glossary																
		2035. Define System Sequence Diagrams																
		2036. Define Operation Contracts																
		2037. Define State Diagrams																
	2040 Design	2041. Define Real Use Cases																
		2042. Define Reports, UI and Storyboards																
		2043. Refine System Architecture																
		2044. Define interaction Class 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																	