OSP Stage 1000

<Plan & Elaboration>

유사도 기반 중심 파일 예측 Clone Checker

Project Team

T4

Date

2016-03-15

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

- Print Manual ‘
- Start Test

- Delete Annotation
- Change Capital
- Detect Change Name
- Detect for To while
- Detect while To for
- Detect if To switch
- Detect switch To if
- Detect Separate Functions
- Detect Combine Functions
- Calculate Part Similarity
- Calculate Total Similarity
- Write Result Text
- Choose 1st Code
- Display 1st Code
- 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
Print Manual	유사도 검사에 앞서 사용자에게 보여줄 매뉴얼을 출력한다
Start Test	유사도 검사를 시작한다.
Delete Annotation	파일의 모든 주석을 삭제한다.
Change Capital	파일의 모든 문자를 대소문자 구분 없이 소문자로 통합한다.
Detect Change Name	변수와 함수 이름을 바꾸었는지 검사한다.
Detect for To while	for 문을 while 문으로 바꾸었는지 검사한다.
Detect while To for	while 문을 for 문으로 바꾸었는지 검사한다.
Detect if To switch	if 문을 switch 문으로 바꾸었는지 검사한다.
Detect switch To if	switch 문을 if 문으로 바꾸었는지 검사한다.
Detect Separate Functions	함수를 고의로 나누었는지 검사한다.
Detect Combine Functions	함수를 고의로 합쳤는지 검사한다.
Calculate Part Similarity	각 단계 검사가 끝난 뒤에 유사도를 백분율(%)로 계산한다.
Calculate Total Similarity	미리 정해 놓은 단계별 가중치에 따라 최종 유사도를 계산한다.
Write Result Text	각 유사도 검사가 끝났을 때 마다 검사 결과를 text 파일로 저장한다.
Compare Similarity	모든 최종 유사도를 비교한다.
Choose 1st Code	가장 유사도가 적은 파일들을 선정한다.
Display 1st Code	가장 유사도가 적은 파일들의 코드를 보여준다.
Exit	유사**도** 검사 프로그램을 종료한다.

Ref. #	Function	Category
R 1.1	Print Manual	Evident
R 1.2	Start	Evident
R 1.2.1	Delete Annotation	Hidden
R 1.2.2	Change Capital	Hidden
R 1.3	Detect Change Name	Hidden
R 1.4	Detect for To while	Hidden
R 1.5	Detect while To for	Hidden
R 1.6	Detect if To switch	Hidden
R 1.7	Detect switch To if	Hidden
R 1.8	Detect Separate Functions	Hidden
R 1.9	Detect Combine Functions	Hidden
R 2.1	Calculate Part Similarity	Hidden
R 2.2	Calculate Total Similarity	Hidden
R 2.2.2	Print Result Text	Hidden
R 3.1	Compare Similarity	Hidden
R 3.2	Choose 1st Code	Hidden
R 3.3	Display 1st Code	Evident
R 4.1	Exit	Evident

2. Operating Environments

OS : Windows

IDE : Eclipse

3. Development Environments

OS : Windows

개발 언어 : JAVA

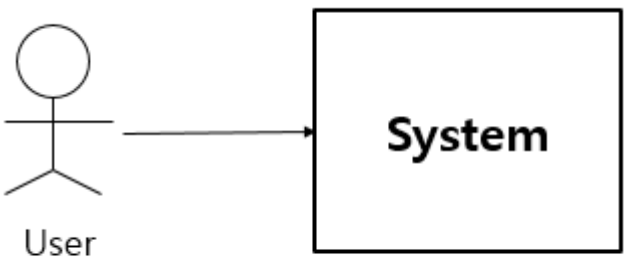
.

Activity1004. Record Terms in Glossary

Term	Description	Remarks
Code	복제된지 확인하려는 코드	
Clone	복제품	
Manual	프로그램의 설명서	
Annotation	주석(메시지나 설명)	
Capital	대문자	
Detect	검사하다	
For	반복문의 한 종류	
While	반복문의 한 종류	
if	조건문의 한 종류	
Switch	조건문의 한 종류	
Separate Functions	나눠진 함수	
Combine Functions	합친 함수	
Part Similarity	단계에 따른 유사도	
Total Similarity	최종의 유사도	
Result Text	유사도 검사의 결과	
Compare Similarity	유사도 데이터의 비교	
1st code	여러 개의 코드들 중 유사도 데이터를 통해 가장 중심이라고 생각되는 코드	
Exit	프로그램 종료	

Activity1006. Define Business Use Case

1. Define System Boundary



2. Identify and Describe Actors

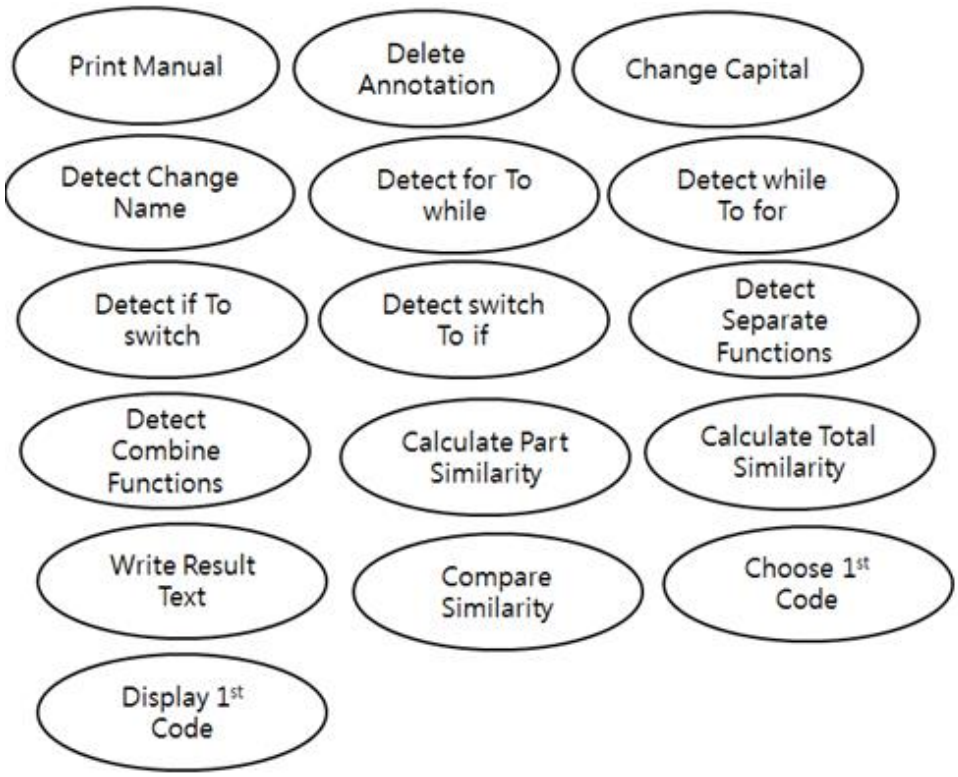
A. User

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

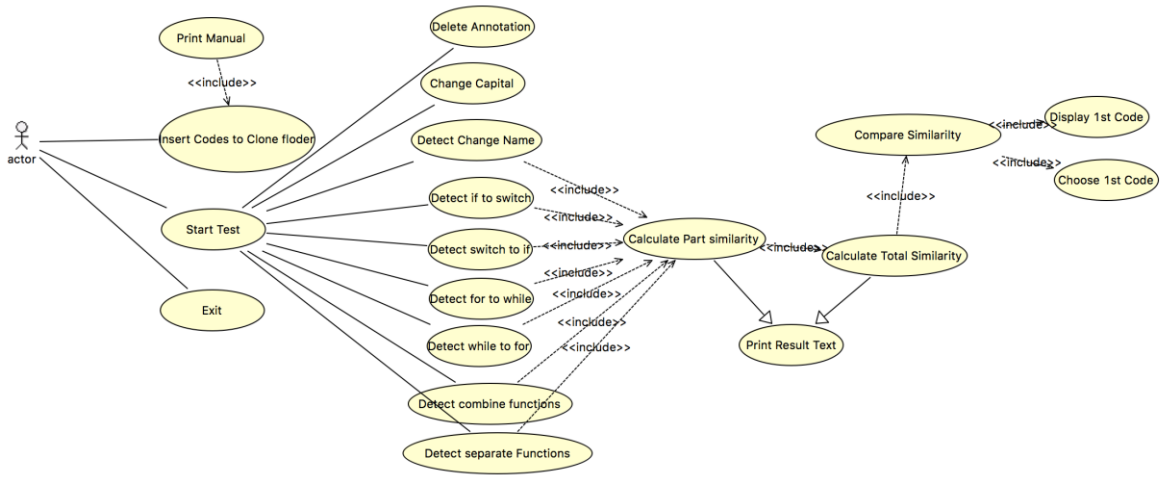
Ref. #	Function	Use-Case
R 1.1	Print Manual	Insert Codes to Clone folder
R 1.2	Start	Start Test
R 1.1	Print Manual	Print Manual
R 1.2.1	Delete Annotation	Delete Annotation
R 1.2.2	Change Capital	Change Capital
R 1.3	Detect Change Name	Detect Change Name
R 1.4	Detect for To while	Detect for To while
R 1.5	Detect while To for	Detect while To for
R 1.6	Detect if To switch	Detect if To switch
R 1.7	Detect switch To if	Detect switch To if
R 1.8	Detect Separate Functions	Detect Separate Functions
R 1.9	Detect Combine Functions	Detect Combine Functions
R 2.1	Calculate Part Similarity	Calculate Part Similarity
R 2.2	Calculate Total Similarity	Calculate Total Similarity
R 2.2.2	Print Result Text	Print Result Text
R 3.1	Compare Similarity	Compare Similarity
R 3.2	Choose 1st Code	Choose 1st Code
R 3.3	Display 1st Code	Display 1st Code
R 4.1	Exit	Exit

5. Categorize Use-Case

Use-Case	Category
Insert Codes to Clone folder	Primary
Start Test	Primary
Print Manual	Primary
Delete Annotation	Primary
Change Capital	Primary
Detect Change Name	Primary
Detect for To while	Primary
Detect while To for	Primary
Detect if To switch	Primary
Detect switch To if	Primary
Detect Separate Functions	Primary

Detect Combine Functions	Primary
Calculate Part Similarity	Primary
Calculate Total Similarity	Primary
Print Result Text	Primary
Compare Similarity	Primary
Choose 1st Code	Primary
Display 1st Code	Primary
Exit	Primary

6. Draw a Use-Case diagram



7. Describe Use-Case

Use Case Name	Insert Codes to Clone folder
Actor	User
Description	시작 전에 검사할 코드들을 정해진 폴더 안에 모아둔다.
Use Case Name	**Start Test**
Actor	User
Description	Clone Checker을 시작한다
Use Case Name	**Print Manual**
Actor	System
Description	시작 전에 해야 할 일들을 알려준다
Use Case Name	**Delete Annotation**
Actor	System

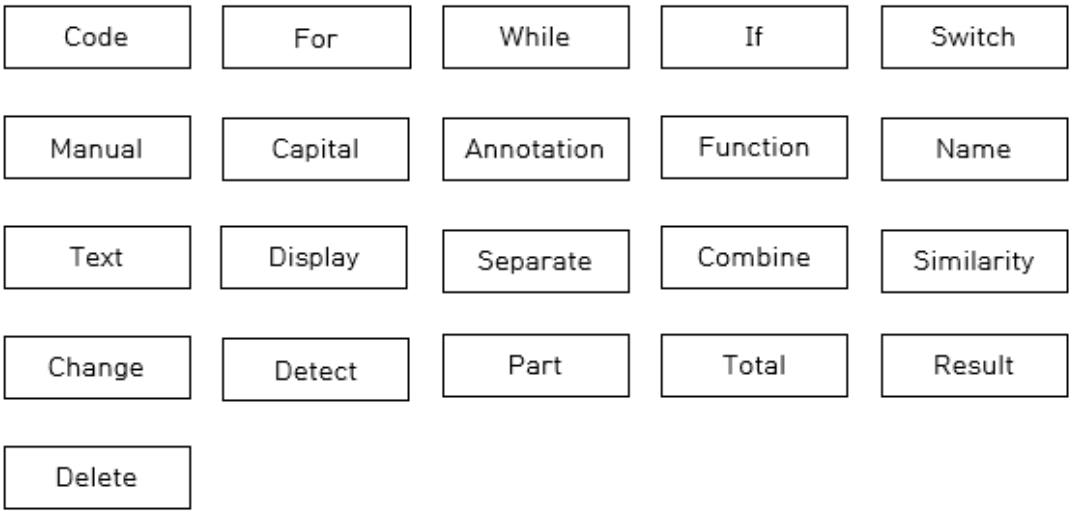
Description	코드에 있는 주석을 제거한다.
Use Case Name	**Change Capital**
Actor	System
Description	코드에 있는 단어들을 모두 소문자로 바꾸어준다.
Use Case Name	**Detect Change Name**
Actor	System
Description	변수이름, 함수이름 이 바뀐 부분을 찾아내준다.
Use Case Name	**Detect for To while**
Actor	System
Description	For 문을 While 문으로 바꾼 곳을 찾아준다.
Use Case Name	**Detect while To for**
Actor	System
Description	While 문을 For 문으로 바꾼 곳을 찾아준다.
Use Case Name	**Detect if To switch**
Actor	System
Description	If 문을 Switch 문으로 바꾼 곳을 찾아준다.
Use Case Name	**Detect switch To if**
Actor	System
Description	Switch 문을 If 문으로 바꾼 곳을 찾아준다
Use Case Name	**Detect Separate Functions**
Actor	System
Description	함수를 의도적으로 분리한 곳을 찾아준다.
Use Case Name	**Detect Combine Functions**
Actor	System
Description	함수를 의도적으로 합친 곳을 찾아준다.
Use Case Name	**Calculate Part Similarity**
Actor	System
Description	단계별로 검사한 유사도를 백분율(%)로 보여준다.
Use Case Name	**Calculate Total Similarity**
Actor	System
Description	단계별로 검사한 유사도를 모두 합산해서 유사도를 보여준다.
Use Case Name	**Print Result Text**
Actor	System
Description	결과를 텍스트 파일로 출력해준다.

Use Case Name	Compare Similarity
Actor	System
Description	모든 최종 유사도를 비교한다.
Use Case Name	**Choose 1st Code**
Actor	System
Description	유사도를 비교해서 원본에 가장 가까운 것 같은 코드를 선택한다.
Use Case Name	**Display 1st Code**
Actor	System
Description	유사도를 비교해서 원본에 가장 가까운 것 같은 코드를 보여준다.
Use Case Name	**Exit**
Actor	A
Description	종료한다.

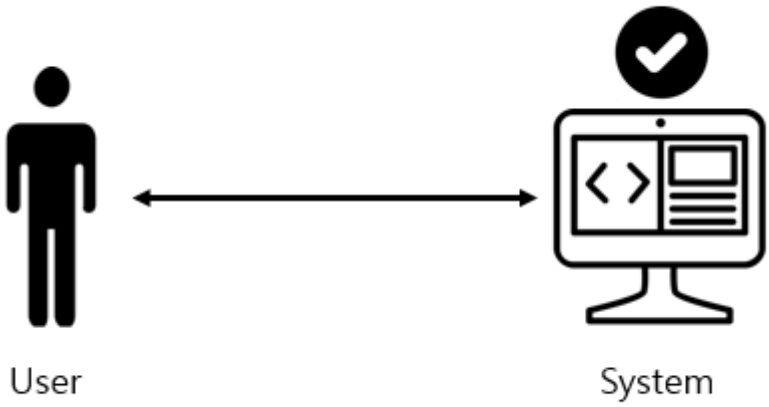
8. Rank Use-Case

Use-Case	Rank
Insert Codes to Clone folder	High
Start Test	High
Print Manual	High
Delete Annotation	High
Change Capital	High
Detect Change Name	High
Detect for To while	High
Detect while To for	High
Detect if To switch	High
Detect switch To if	High
Detect Separate Functions	High
Detect Combine Functions	High
Calculate Part Similarity	High
Calculate Total Similarity	High
Print Result Text	High
Compare Similarity	High
Choose 1st Code	High
Display 1st Code	High
Exit	High

Activity1007. Define Business Concept Model



Activity1008. Define Draft System Architecture



Activity1009. Define System Test Case

Identifier	Feature
CK.STC.100	Clone 폴더에 검사할 모든 c 파일을 넣으라는 말이 나오는지 검사한다.
CK.STC.101	Clone 폴더의 파일 목록이 제대로 받아졌는지 검사한다.
CK.STC.102	파일 내의 모든 주석이 지워졌는지 검사한다.
CK.STC.200	파일 내의 모든 문자가 소문자로 바뀌었는지 검사한다.
CK.STC.201	변수 명을 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.202	함수 명을 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.203	if 문을 switch 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.204	switch 문을 if 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.205	for 문을 while 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.206	while 문을 for 문으로 바꾼 코드를 잘 찾아 냈는지 검사한다.
CK.STC.207	함수를 고의적으로 나눈 코드를 잘 찾아 냈는지 검사한다.
CK.STC.208	함수를 고의적으로 합친 코드를 잘 찾아 냈는지 검사한다.
CK.STC.300	각 단계별 유사도 백분율이 알맞게 계산이 되었는지 검사한다.
CK.STC.301	단계별 가중치를 적용한 유사도 백분율이 알맞게 계산이 되었는지 검사한다.
CK.STC.302	유사도 검사가 끝났을 때 마다 결과를 작성한 텍스트 파일이 맞게 생성되었는지 검사한다.
CK.STC.303	유사도 검사를 통하여 코드들 중 가장 중심이 될 것이다 예상되는 코드를 알맞게 정했는지 검사한다.
CK.STC.304	유사도 검사를 통하여 코드들 중 가장 중심이 될 것이다 예상되는 코드를 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

- - Print Manual ‘
- Start Test
- Delete Annotation
- Change Capital
- Detect Change Name
- Detect for To while
- Detect while To for

- Detect if To switch
- Detect switch To if
- Detect Separate Functions
- Detect Combine Functions
- Calculate Part Similarity
- Calculate Total Similarity
- Write Result Text
- Choose 1st Code
- Display 1st Code
- Exit

4. Performance Requirements

- 단계별 유사도 검사 속도가 빨라야 한다.

5. Operating Environment

- OS : Window
- IDE : Eclipse

6. User Interface Requirements

7. Other Requirements

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

8. Resources

- Human Resource : 3 명

- Project Duration : 16 주

- Cost : 100 만원

9. 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										