Software Modeling & Analysis

Functional Digital Watch

Response Report #2

Title

용사여, 일어나시계

Date

2019-06-10

Team 4

201511246 김상재

201511272 양재민

201511292 전도현

201710515 최연지

Index

1. Specification Review

1.1 Stage 1000 Planning

◆ Activity 1006. Define Business Use Case

- 5. Use Case Diagram
  - Set Summer Time 제거 완료

1.2 Stage 2030 Analysis

◆ Activity 2031. Define Essential Use Cases

- 22. Show Alarm
  - ➔ Show realtime 을 Change Type으로 수정완료
- 23. Set Alarm Time
  - ➔ Bell section을 벨소리 section으로 수정 완료.
- 31. Show Sun
  - ➔ Sun모드에 대한 설명 추가 완료

1.3 Stage 2040 Design

➔ Stage 2030 Analysis 단계와 동일하게 수정완료

2. PMD Result - Bugs

2.1 Bug #1

```
-       try { Thread.sleep(3000);} catch(InterruptedException e) {e.printStackTrace();}
+
+       /* [sonarqube][Bug #1] */
+       try { Thread.sleep(3000);} catch(InterruptedException e) {
+           e.printStackTrace();
+           Thread.currentThread().interrupt();
+       }
+
```

Bug #2

```
-       try { Thread.sleep(3000);} catch(InterruptedException e) {e.printStackTrace();}
+
+       /* [sonarqube][Bug #2] */
+       try { Thread.sleep(3000);} catch(InterruptedException e) {
+           e.printStackTrace();
+           Thread.currentThread().interrupt();
+       }
```

Bug #3

```
-       while(true) {
-           try {
                sleep(10);
-           } catch (InterruptedException e) {
-               e.printStackTrace();
            }
-           system.realTimeTask();
+       try {
+           while(true) {
                sleep(10);
+               system.realTimeTask();
            }
+       } catch (InterruptedException e) {
+           //e.printStackTrace();
+           /* [sonarqube][Bug #3] */
+           Thread.currentThread().interrupt();
+           return;
        }
```

- src/main/java/Alarm.java → "interruptedException" should not be ignored

  ➔ Thread.interrupt()를 호출해서 re-interrupted로 해결

2.2 Bug #4

```
-       if(this.currTime.getTimeInMillis() >= this.sun[1].getTimeInMillis()){
+
+       /* [sonarqube][Bug #4] */
+       if(this.currTime.getTimeInMillis() >= this.getSun(1).getTimeInMillis()){
```

Bug #5

```
-       if(this.currTime.getTimeInMillis() > this.sun[0].getTimeInMillis()){
```

```
-       if(this.currTime.getTimeInMillis() > this.sun[1].getTimeInMillis()){
+       /* [sonarqube][Bug #5] */
+       if(this.currTime.getTimeInMillis() > this.getSun(0).getTimeInMillis()){
+       /* [sonarqube][Bug #5] */
+       if(this.currTime.getTimeInMillis() > this.getSun(1).getTimeInMillis()){
```

- src/main/java/Alarm.java → Method accesses list of array with constant index

  ➔ 배열에 직접 접근이 아닌 접근자(get)를 이용하는걸로 대체

2.3 Bug #6

```
-       TimeThread timeThread = new TimeThread(watchSystem);
-       timeThread.run();
+       /* [sonarqube][Bug #6] */
+       Thread timeThread = new TimeThread(watchSystem);
+       timeThread.start();
```

Bug #7

```
-       this.run();
```

```
+       /* [sonarqube][Bug #7] */
+       if(this.getState() == State.NEW)
+           this.start();
```

- src/main/java/Main.java → "Thread.run()" should not be called directly

➔ Thread.run()을 직접 호출하지 않고 Thread.start() 하는 것으로 대체

2.3 Bug #8

```
-       this.currTime = this.realTime.requestRealTime();
+
+       /* [sonarqube][Bug #8] */
+       Calendar currTime = this.realTime.requestRealTime();
+       //this.currTime = this.realTime.requestRealTime();
```

```
-               (this.currTime.get(Calendar.HOUR_OF_DAY) == this.alarm[i].get(Calendar.HOUR_OF_DAY)) &&
-               (this.currTime.get(Calendar.MINUTE) == this.alarm[i].get(Calendar.MINUTE)) &&
-               (this.currTime.get(Calendar.SECOND) == this.alarm[i].get(Calendar.SECOND))
+               (currTime.get(Calendar.HOUR_OF_DAY) == this.alarm[i].get(Calendar.HOUR_OF_DAY)) &&
+               (currTime.get(Calendar.MINUTE) == this.alarm[i].get(Calendar.MINUTE)) &&
+               (currTime.get(Calendar.SECOND) == this.alarm[i].get(Calendar.SECOND))
```

```
-          if(this.status == 4 && (this.currTime.get(Calendar.SECOND) == this.alarm[i].get(Calendar.SECOND) + 30)) {
```

```
+          if(this.status == 4 && (currTime.get(Calendar.SECOND) == this.alarm[i].get(Calendar.SECOND) + 30)) {
```

- src/main/java/Alarm.java → Class defineds fields that are used only as locals

  ➔ 객체변수를 지역변수로 바꿔서 해결

2.4 Bug #9 (해결 불가)

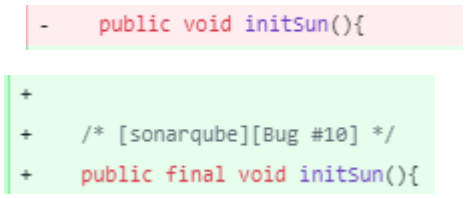
```
-    public WatchSystem() throws UnsupportedAudioFileException, IOException, LineUnavailableException{
+    public WatchSystem(){
```

```
-       menu.add(new Timer());
-       menu.add(new Alarm((RealTime)menu.get(1)));
+
+       /* [sonarqube][Bug #9] */
+       try{
+
+           menu.add(new Timer());
+           menu.add(new Alarm((RealTime)menu.get(1)));
+       }
+       catch(Exception e) {
+           e.printStackTrace();
+           return;
+       }
```

- src/main/java/WatchSystem.java → Class has a circular dependency with other classes

  ➔ 해당 코드를 해결하는 것 문제가 아니었으며 WatchSystem과 WatchGUI의 Circular Dependency를 해결하려 했으나 WatchSystem에 대한 Interface를 만들어 선언하거나, WatchGUI가 ActionListener를 interface로 활용하여 만들어진 클래스여서 각 선언부의 클래스 명을 ImpleSystem (WatchSystem을 위하여 만든 Empty Interface)와 ActionListener를 사용하여 선언을 하였으나 해결이 되질 않았다.

2.5 Bug #10



- src/main/java/WatchSystem.java → Constructor makes call to non-final method

  ➔ 생성자에서 실행시키는 메소드는 final 메소드 여야 하기 때문에 initSun()메소드를 final 메소드로 변경함

3. PMD Result - Vulnerabilities

3.1 Vulnerabilities #1

```
-        return displayModeData;
```

```
+
+        /* [sonarqube][Vuln #1] */
+        return displayModeData.clone();
```

3.1 Vulnerabilities #2

```
-        return displayRealTimeData;
```

```
+
+        /* [sonarqube][Vuln #2] */
+        return displayRealTimeData.clone();
```

3.1 Vulnerabilities #3

```
-            return displaySettingTimeData;
```

```
+
+            /* [sonarqube][Vuln #3] */
+            return displaySettingTimeData.clone();
```

3.1 Vulnerabilities #4

```
-        return displaySWData;
```

```
+
+        /* [sonarqube][Vuln #4] */
+        return displaySWData.clone();
```

3.1 Vulnerabilities #5

```
-        return displaySunDisplay;
```

```
+        /* [sonarqube][Vuln #5] */
+        return displaySunDisplay.clone();
```

3.1 Vulnerabilities #6

```
-        return displayTimerData;
```

```
+
+        /* [sonarqube][Vuln #6] */
+        return displayTimerData.clone();
```

- 배열을 사용하는 곳에서 나타남 → Security – Method returns internal array
  - ➔ .clone()을 통해 내부 배열을 사용자가 수정할 수 없도록 복사본을 반환

3.2 Vulnerabilities #14

```
System.out.println(getClass().getResource("").getPath()+"sounds/Alarm"+index+".wav");
audioInputStream = AudioSystem.getAudioInputStream(new File(getClass().getResource("").getPath()+"sounds/Alarm"+index+".wav"));

e.printStackTrace();
```

- src/main/java/Bell.java

3.2 Vulnerabilities #15

```
try {
    this.mainFont = Font.createFont(Font.TRUETYPE_FONT, new File(WatchSystem.class.getResource("").getPath() + "DS-DIGI.TTF"));
} catch (FontFormatException ex) {
,8 @@ public class WatchGUI implements ActionListener {
} catch (IOException ex) {
    ex.printStackTrace();
}
e.printStackTrace();

e.printStackTrace();
/* [sonarqube][Vuln #15] */
// IDE Test
/*
try {
    this.mainFont = Font.createFont(Font.TRUETYPE_FONT, new File(WatchSystem.class.getResource("").getPath() + "DS-DIGI.TTF"));
} catch (FontFormatException ex) {
,8 @@ public class WatchGUI implements ActionListener {
} catch (IOException ex) {
    ex.printStackTrace();
}
```

- src/main/java/WatchGUI

  → Handling files is security-sensitive

  - ➔ IDE에서 실행할 때는 파일 경로를 string으로 하여 파일을 접근 해야했지만, 그 방식이 안전하지 않다 하여 제거함.

3.3 Minor Vulnerabilities #1

```
            e.printStackTrace();
+           /* [sonarqube][Use a logger to log this exception] */
+           Logger logger = Logger.getLogger(Bell.class.getName());
+           logger.log(Level.ALL, "[Bell] Exception", e);
+           Thread.currentThread().interrupt();
```

- src/main/java/Alarm.java
- src/main/java/Bell.java
- src/main/java/WatchGUI.java
- src/main/java/WatchSystem.java

→ Use a logger to log this exception

➔ e.printStackTrace()를 Logger Class를 사용하여 Log를 출력하게함.

3.3 Minor Vulnerabilities #2

```
-    public ArrayList<String> getNewMode() { return newMode; }
-    public void setNewMode(ArrayList<String> newMode) { this.newMode = newMode; }
-    public void setOldMode(ArrayList oldMode) { this.prevModeObject = oldMode; }
-    public ArrayList getPrevModeObject(){ return this.prevModeObject; }
+    /* [sonarqube][Return a copy of] */
+    public ArrayList<String> getNewMode() { return (ArrayList)newMode.clone(); }
+
+    /* [sonarqube][Return a copy of] */
+    public void setNewMode(ArrayList<String> newMode) { this.newMode = (ArrayList)newMode.clone(); }
+
+    /* [sonarqube][Store a copy of] */
+    //public void setOldMode(ArrayList oldMode) { this.prevModeObject = (ArrayList)oldMode.clone(); }
+
+    /* [sonarqube][Store a copy of] */
+    public ArrayList getPrevModeObject(){ return (ArrayList)this.prevModeObject.clone(); }
```

- src/main/java/ModeSetting.java
- src/main/java/WatchSystem.java

→ Store copy of "newMode", Return a copy of "newMode"

➔ .clone()을 통해 내부 배열을 사용자가 수정할 수 없도록 복사본을 반환

3.3 Minor Vulnerabilities #3

```
-    public JTextField[] showDate = new JTextField[]{new JTextField(), new JTextField(), new JTextField()};
+    private JTextField[] showDate = new JTextField[]{new JTextField(), new JTextField(), new JTextField()};
```

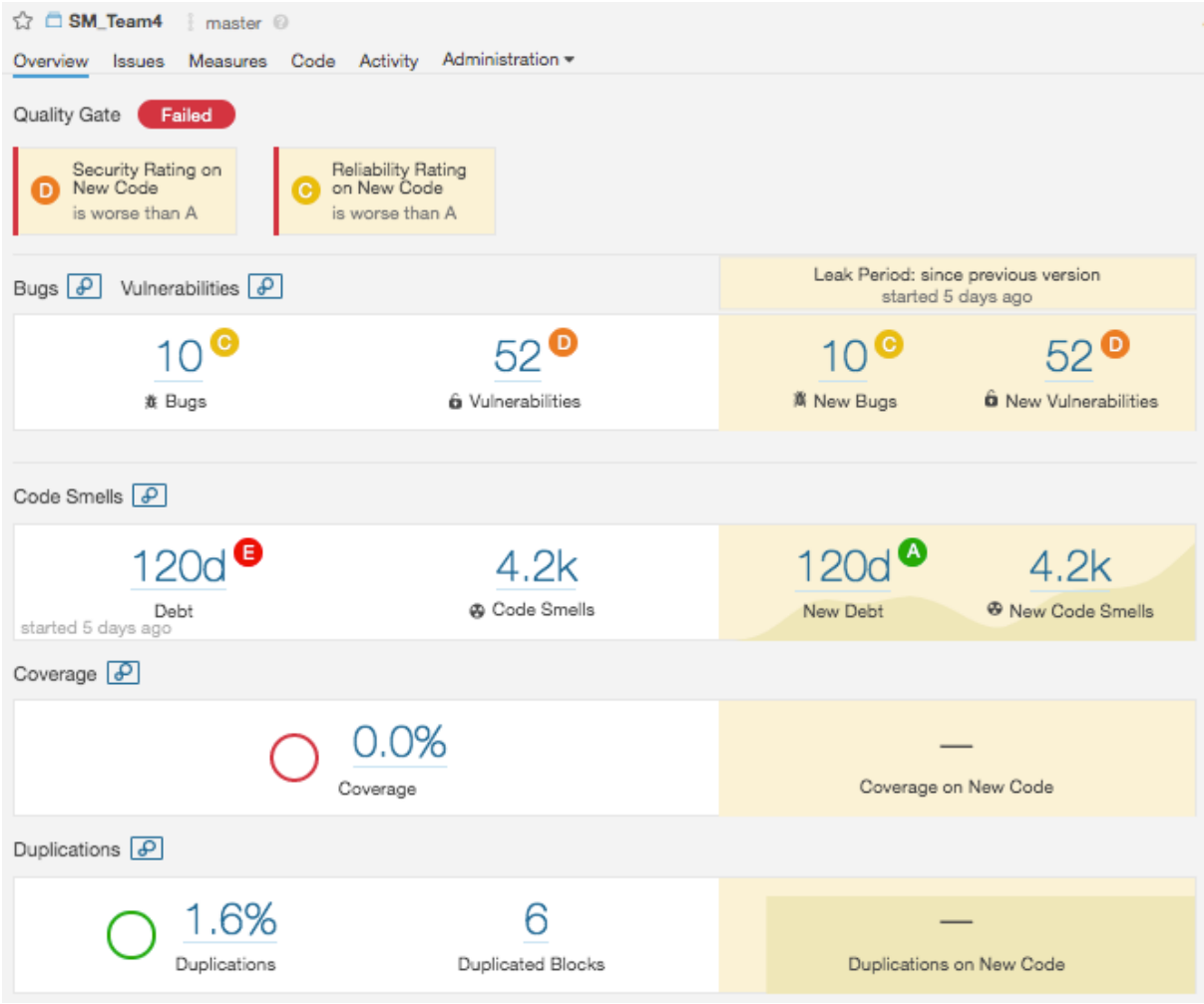
- src/main/java/Main.java
- src/main/java/WatchGUI.java

  → public -> private

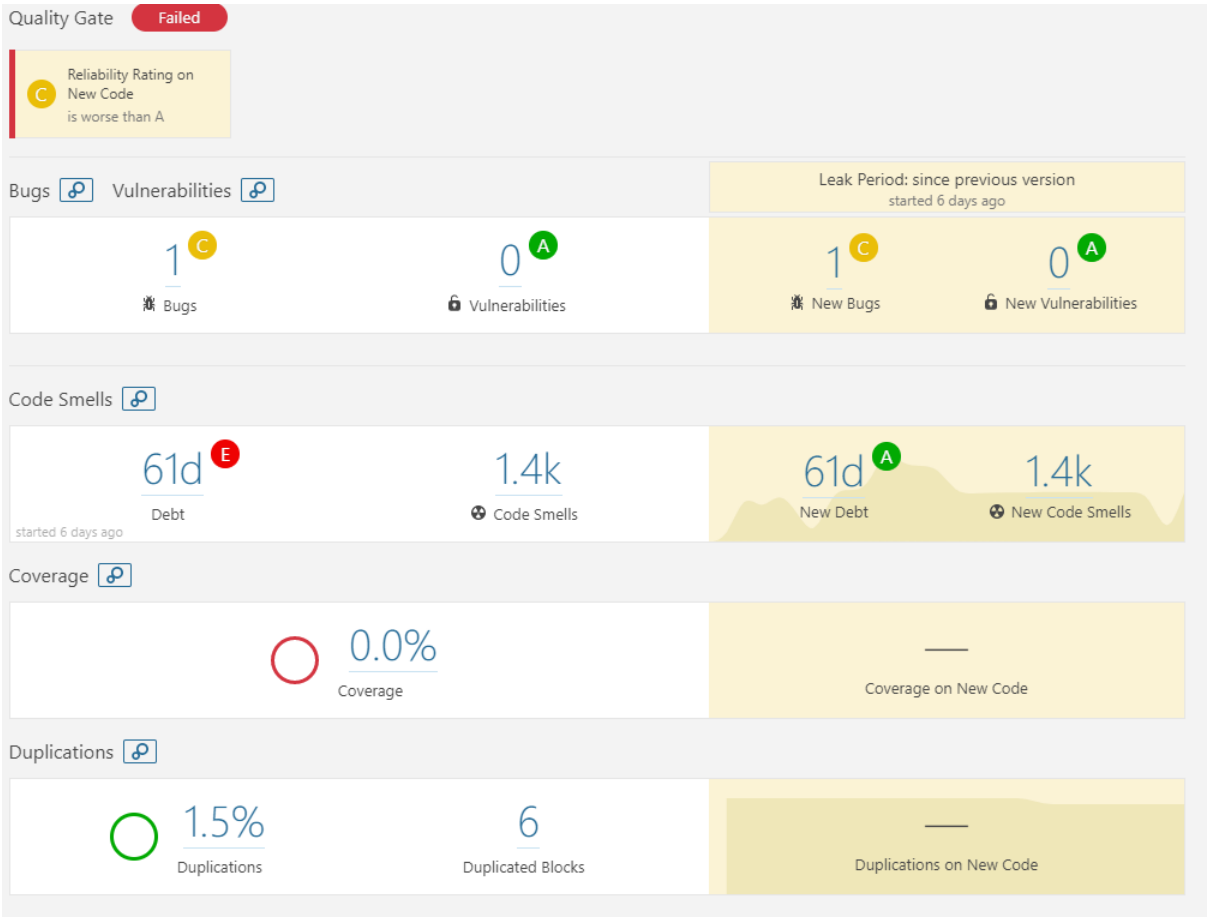
  ➔ public을 private로 변경하여 해결

3.4 Before & After

- Before



- After


Quality Gate
Failed
C
Reliability Rating on New Code
is worse than A
Bugs
Vulnerabilities
Leak Period: since previous version
started 6 days ago
1
C
Bugs
0
A
Vulnerabilities
1
C
New Bugs
0
A
New Vulnerabilities
Code Smells
61d
E
Debt
started 6 days ago
1.4k
Code Smells
61d
A
New Debt
1.4k
New Code Smells
Coverage
0.0%
Coverage
—
Coverage on New Code
Duplications
1.5%
Duplications
6
Duplicated Blocks
—
Duplications on New Code

4. Findbugs Result

4.1 Code Smell #1

- Return value: call by reference
  - src/main/java/Alarm.java

```
return displayAlarmData;
/* [sonarqube][Alarm.showAlarm() may expose internal representation by returning displayAlarmData] */
return displayAlarmData.clone();
```

  - src/main/java/ModeSetting.java

```
return displaySettingTimeData;

/* [sonarqube][Vuln #3] */
return displaySettingTimeData.clone();
```

  - src/main/java/RealTime.java

```
return displayRealTimeData;

/* [sonarqube][Vuln #2] */
return displayRealTimeData.clone();
```

  - src/main/java/Stopwatch.java

```
  return displaySWData;

  /* [sonarqube][Vuln #4] */
  return displaySWData.clone();
```

  - src/main/java/Sun.java

```
return displaySunDisplay;
/* [sonarqube][Vuln #5] */
return displaySunDisplay.clone();
```

  - src/main/java/Timer.java

```
return displayTimerData;

/* [sonarqube][Vuln #6] */
return displayTimerData.clone();
```

  - src/main/java/WorldTime.java

```
return displayWTData;
return displayWTData.clone();
```

4.1 Code Smell #2

- Storing externally mutable object
  - src/main/java/Alarm.java

```
public void setReservated(Calendar[] reservated) { this.reservedAlarm = reservated; }
/* [sonarqube][Alarm.setReservated(Calendar[]) may expose internal representation by storing an externally mutable object into reservedAlarm] */
public void setReservated(Calendar[] reservated) { this.reservedAlarm = reservated.clone(); }
```

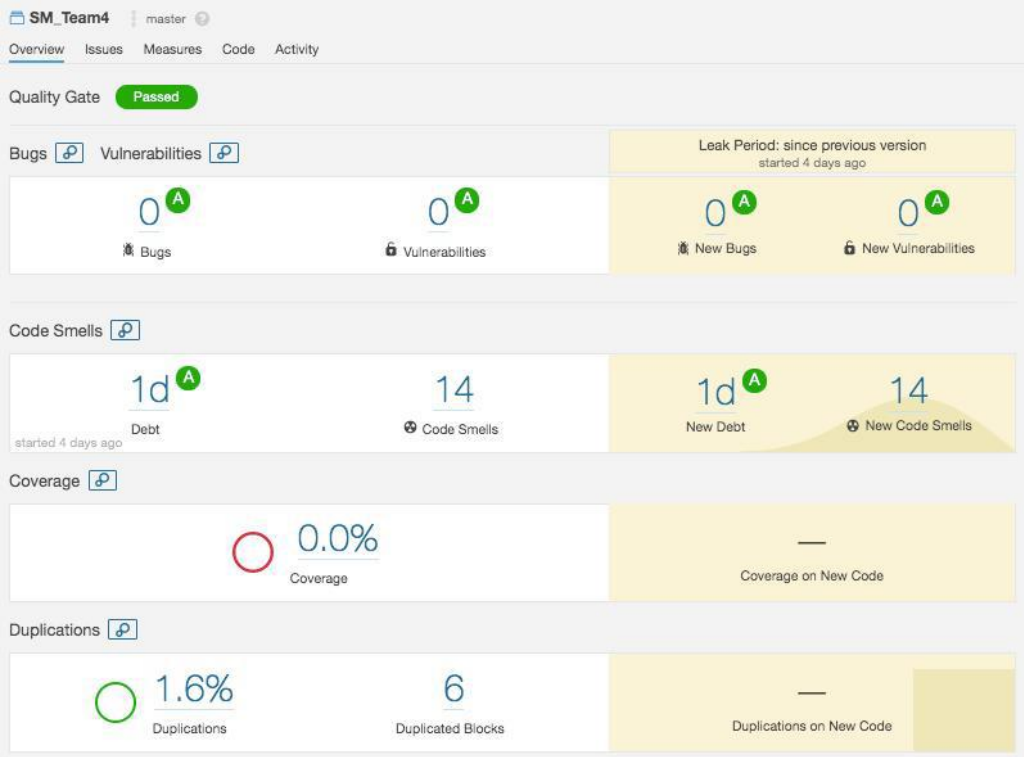
4.1 Code Smell #3

- Default case missing
  - src/main/java/Alarm.java

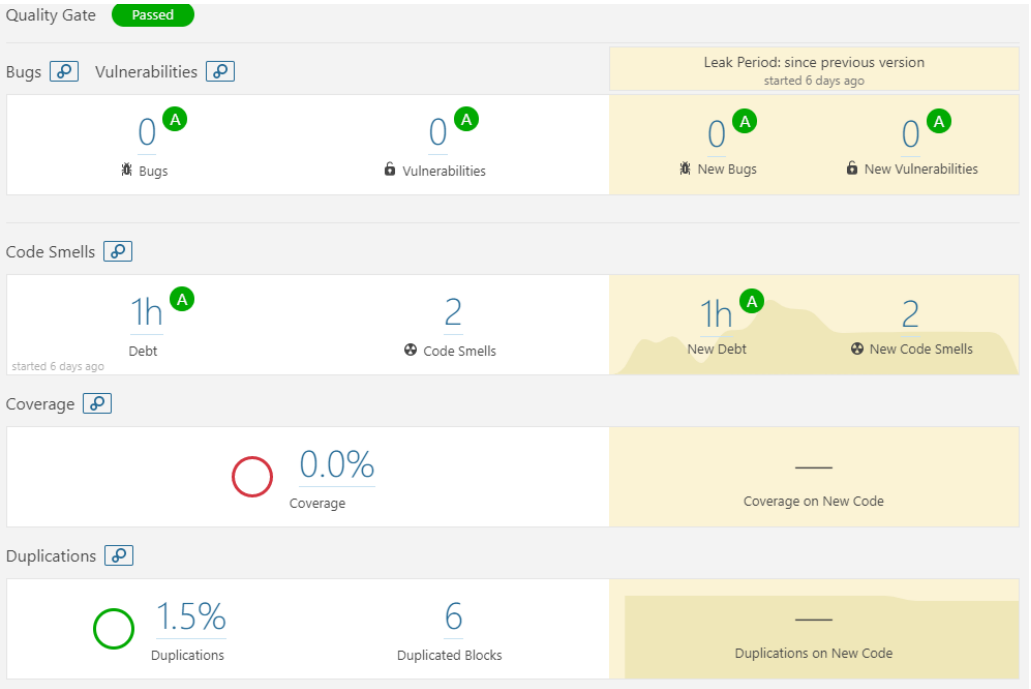
```
switch (currSection) {
    case 0: displayAlarmData[3] = ""; break;
    case 1: displayAlarmData[6] = ""; break;
,31 @@ public class Alarm {
    case 3: displayAlarmData[1] = ""; break;
    case 4: displayAlarmData[0] = " "; break;
    case 5: displayAlarmData[4] = ""; break;
    default: break;
```

4.2 Before & After

- Before



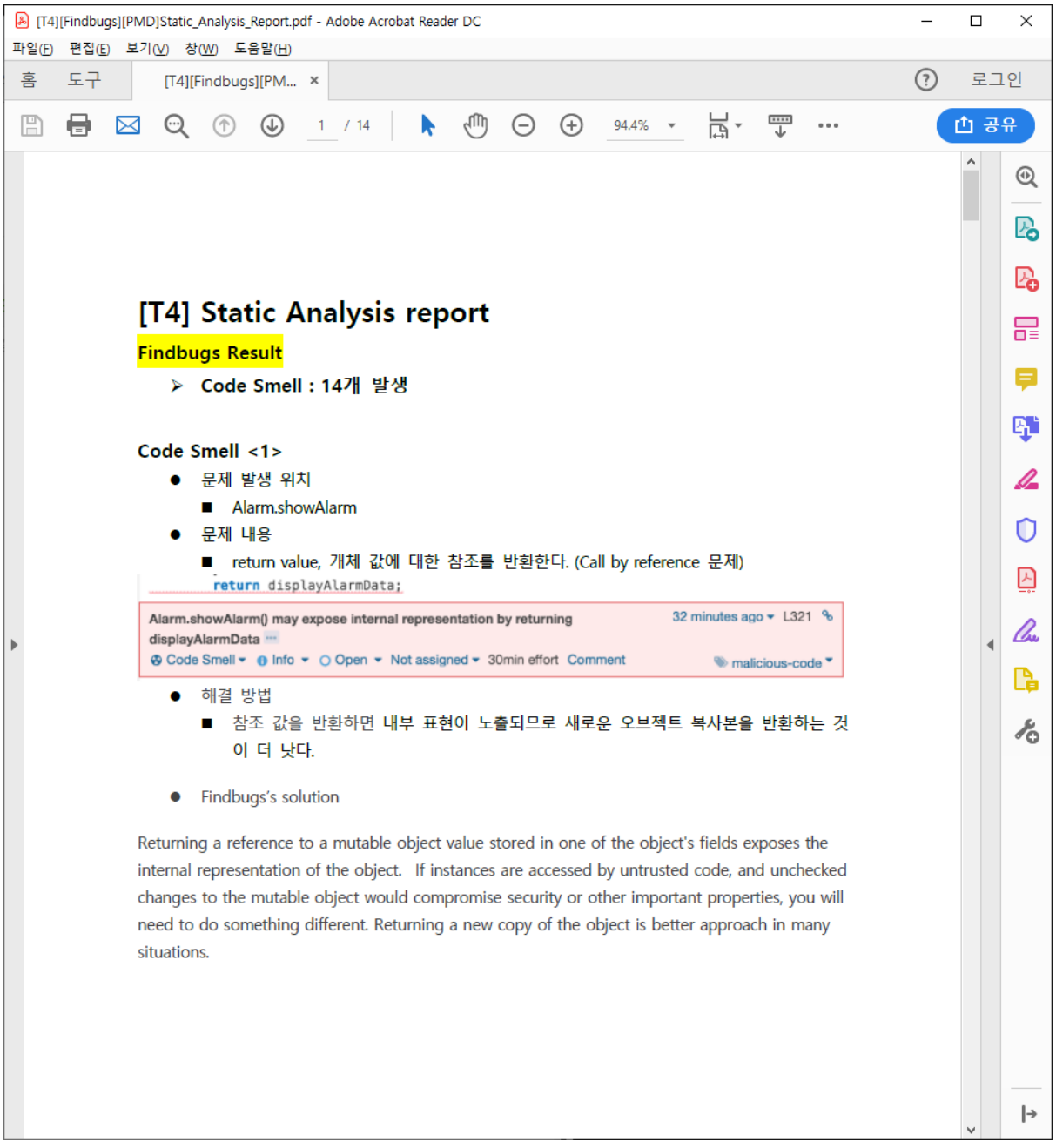
- After

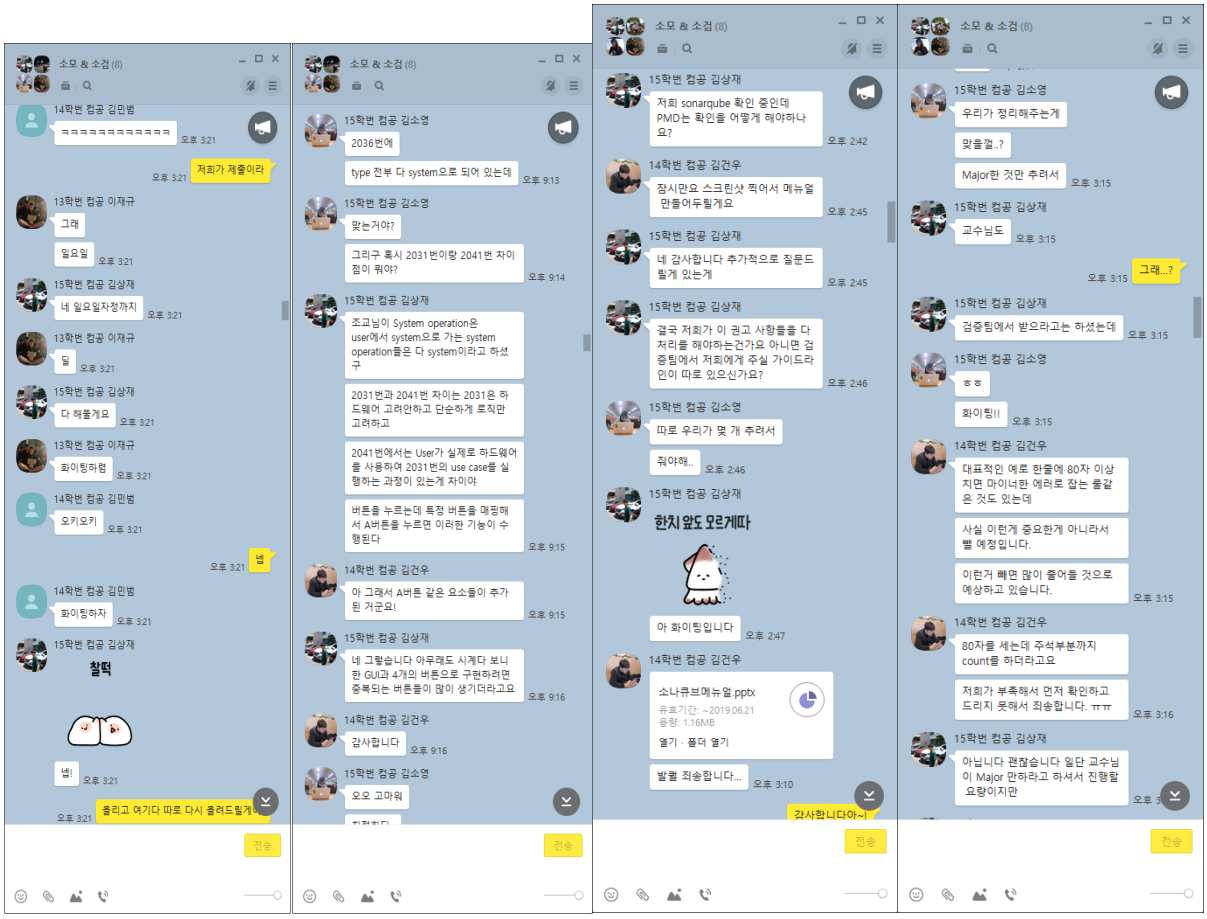


5. Closing

5.1 Software Testing Team Cooperation

- 이전 보고서에 대해 부족한 부분들에 대한 상호 보완 및 협의
- 추가적인 보고서 자체 제작, 공유





5.2 Class Review

김상재: 이번 모델링 수업을 계기로 비록 큰 시스템에 구동되는 소프트웨어가 아닌 작은 시스템에서 구동되는 소프트웨어를 제작하였지만 이 OOAD 과정을 거침으로서 어느 단계에서 어떻게 생각을 해야 후에 고생을 안 하는 지 뼈저리게 느꼈습니다. 처음부터 신중하고 정확하게 해야 다음 단계가 수월하게 이루어짐을 알 수 있었습니다.

양재민: 한 학기동안 정말 힘들었지만 그만큼 배우는 게 많았던 수업이었습니다. 소프트웨어 모델링을 하는데 잘 배울 수 있는 기회였고 많은 것을 얻어가는 수업이었습니다. 하지만, 다른 과제나 프로젝트도 하다 보니 일정이 많이 타이트해서 많이 힘들었습니다.

전도현: 단순 개발 프로젝트가 아닌 기획부터 분석, 디자인 단계를 직접 경험하였고, 그를 기반으로 개발을 하고, 4학년 수업과 연계되어 검증까지 받으면서 많은 것들을 배울 수 있었습니다. 각 단계들을 거치며 일주일이 멀도록 성장한 것 같아 뿌듯했습니다. 물론 그만큼 고생도 많이 했지만, 초기 단계의 중요성을 몸소 깨달을 수 있었습니다. 코딩이 중요한 게 아니라 이를 설계해주는 역할의 중요성을 깨닫고, 좀 더 큰 그림과 큰 배경을 볼 수 있게 된 것 같아 이 수업을 들은 것이 정말 좋은 선택이었음을 확신합니다. 또한 검증팀과의 협업이 매우 잘 이루어져 상호간의 피드백을 받으며, 만든 것을 본 사람들이 이해할 수 없는 부분들이 있다는 것을 배웠고, 팀 내에서

의 불화나 스타일 등에 어려움을 겪으며 PM의 중요성도 제대로 알 수 있었습니다. 비록 4명밖에 안되는 팀에서 얼마나 다르면 다르겠냐고 할 수도 있었지만, 같은 말을 표현하는 방식, 생각하는 방식에 있어 부딪히는 부분에 있어 서로 합의하고 공감하는 능력도 많이 기를 수 있었습니다.

최연지: 우리 학교만의 자랑인 소프트웨어 모델링 수업을 들을 수 있어서 뿌듯했습니다. 다른 과목에선 해볼 수 없는 경험들이었고 졸업하는 데에 큰 도움이 될 것 같고, 졸업하고 나서도 큰 도움이 될 수업이었습니다. 그리고 검증 수업은 절대 듣지 말아야지라고 생각했습니다. 진로를 전향할까 진지하게 고민해보는 계기가 된 수업이었습니다. 소모 덕분에 제 한학기가 삭제된 것 같아서 좋아해야 할지 모르겠습니다. 바쁘게 살고 싶은 후배들에게 꼭 이 수업을 추천하겠습니다.