Software Modeling & Analysis

Traveler Digital Watch

Project Team
7 Team

Date
2019-06-09

Team Information
201411295 이상훈
201711394 민하은
201711395 박성준
201711423 정종화

Index

Activity 1. Requirement Revision

- 1.1 Stage 1000 Planning
- 1.2 Stage 2030 Analysis
- 1.3 Stage 2040 Design
- 1.4 Stage 2050 Implementation
- 1.5 Stage 2060 Unit test

Activity 2. System Test Response

Activity 3. Static Analysis Response

Activity 4. OOPT Review

Activity 1. Requirement Revision

1.1 Stage 1000 Planning

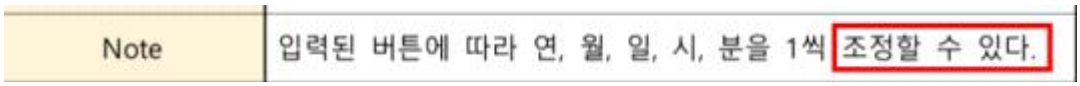
1.2 Stage 2030 Analysis

1.3 Stage 2040 Design

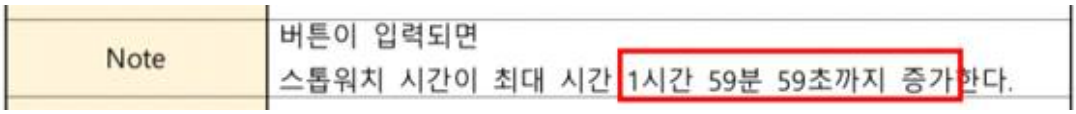
R.2.1 Start Timer
- Problem
1. (A) 타이머 시작버튼을 누른다. 2. (S) 타이머에 저장된 시간을 1초씩 줄이며 흐르게 한다. 3. (S) 타이머 시간이 0이 되면 알람을 10초 울린다. 4. (S) 타이머 시간 화면에 초과된 시간을 1초씩 흐르게 한다. - Timer가 종료된 시점에서도 알람이 울리지 않는다. 소리가 나지 않을 뿐만 아니라 알람 표시조차 없다.
- Fixed
 - Timer가 종료되면 알람이 울리도록 변경하고, 알람 표시가 나도록 수정.

R.2.1 Start Timer
- Problem
- 시계의 초에 해당하는 부분이 invalid 한 값이 등장한다. (00 ~ 59가 아닌 74가 나오는 문제 발생.
- Fixed
- 시계 범위에 맞는 값이 나오도록 조정 : 00 ~ 59

1.4 Stage 2050 Implementation

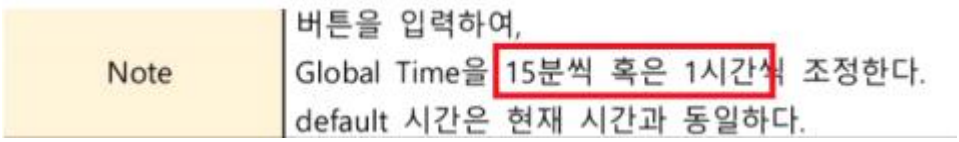
R.1.0 Set Time
- Problem
 - 시계의 현재 시간 조정 안됨. 조정하고 싶은 시간으로 맞춘 뒤, 결과가 저장되지 않는다.
- Fixed(x) -> 문제없음
- Set time에서 조정하고 싶은 시간으로 맞춘 뒤, 저장 버튼을 누른 후 set Time 모드를 종료시켜야 결과가 저장됨. 저장되지 않은 경우, 조정된 시간을 무시하고 처음에 설정 되어 있던 time을 보여주므로 저장 후 mode를 종료시켜야 한다.

R.2.1 Start Timer
- Problem
- 타이머 모드 초기 상태인 00:00:00에서 시작 버튼을 누를 경우 프로그램이 먹통이 되어버림
- Fixed
- 타이머 모드 초기 상태에서 시작 버튼을 눌러도 특별한 문제가 발생하지 않도록 수정

R.3.0 Start Stopwatch
- Problem
 - 스톱워치 시간이 59분 59초인 상황에서 1시간으로 넘어가지 않는다. - 즉 시간의 단위가 올라가지 않는다.
- Fixed
- 최대 2시간까지 가능하며 02:00:00에서 멈춘다.

R.5.2 Set Number Range
- Problem
 - 난수의 범위를 무한히 정할 수 있는 것이 아니라 최댓값이 59로 정해져있다. - 해당 설명이 누락되어 있다.
- Fixed
- 설명이 명확하도록 수정하였다. -> 생성될 난수의 범위를 1부터 최대 59까지 조정한다.

R.5.2 Set Number Range
- Problem
 - 난수 생성 범위가 0이 아닌 1로 초기화 된다.
- Fixed
- 설명을 올바르게 수정하였다. -> 난수 생성 범위를 1로 초기화 한다.

R.1.15 Set Global Time
- Problem
 - 1번 누를 시 15분, 2번 누를 시 30분, 3번 누를 시 45분, 이렇게 변하는 것이 아니라 1번 누를 시 15분, 2번 누를 시 45분, 3번 누를 시 1시간 30분, 이렇게 차이값이 15분씩 올라감.
- Fixed
- 설명이 명확하도록 수정하였다. -> 시간을 조정할 때 분은 15분-> 30분 -> 45분 ..씩 증가한다. -> 시간은 1시간 -> 2시간-> 3시간...씩 증가한다. - 시간이 올바르게 올라가도록 수정하였다.

1.5 Stage 2060 Unit Test

Activity 2. System Test Response

1. Set time 모드에서 유효한 시간으로 변경 후 오른쪽 위 버튼을 길게 눌러 바꾼 시간을 적용한다.
- Problem
- 시간을 바꿀 수는 있으나 오른쪽 위 버튼을 길게 눌러도 바꾼 시간이 정상적으로 적용되지 않음.
- Fixed(x) -> 문제없음
- Set time에서 조정하고 싶은 시간으로 맞춘 뒤, 저장 버튼을 누른 후 set Time 모드를 종료시켜야 결과가 저장됨. 저장되지 않은 경우, 조정된 시간을 무시하고 처음에 설정 되어 있던 time을 보여주므로 저장 후 mode를 종료시켜야 한다.

2. Global Time모드에서 왼쪽 위 버튼을 짧게 눌러 Global Time을 15분 씩 증가시킨다.

- Problem

- 시간을 증가시킬 수는 있으나 '15분 -> 30분 -> 45분'이 아닌 '15분 -> 45분 -> 1시간 30분'으로 증가함. 즉 더해지는 폭이 15분씩 증가함

- Fixed

```
public GlobalTime() {
    this.gTime = super.time;
    this.modifiedMinute = 15;
    this.modifiedHour = 1;
    this.simpleDateFormat = new SimpleDateFormat( pattern: "yy-MM-dd HH:mm:ss");
    this.stringGTime = this.simpleDateFormat.format(this.gTime.getTime());
    this.timer.scheduleAtFixedRate(this.task,  delay: 0L,  period: 1000L);
}
```

```
public void saveMinutesAdd() { this.gTime.add( field: 12, this.modifiedMinute); }
```

- modifiedMinute 변수를 통해 분이 단위에 맞게 증가할 수 있도록 수정하였다.

3. Global Time 모드에서 왼쪽 위 버튼을 길게 눌러 Global Time을 1시간 씩 계속 증가시킨다.

- Problem

- 시간을 증가시킬 수는 있으나 '1시간 -> 2시간-> 3시간'이 아닌 '1시간 -> 3시간 -> 6시간' 으로 증가함. 즉 더해지는 폭이 1시간씩 증가함

- Fixed

```
public GlobalTime() {
    this.gTime = super.time;
    this.modifiedMinute = 15;
    this.modifiedHour = 1;
    this.simpleDateFormat = new SimpleDateFormat( pattern: "yy-MM-dd HH:mm:ss");
    this.stringGTime = this.simpleDateFormat.format(this.gTime.getTime());
    this.timer.scheduleAtFixedRate(this.task,  delay: 0L,  period: 1000L);
}
```

```
public void saveHoursAdd() { this.gTime.add( field: 10, this.modifiedHour); }
```

- modifiedHour 변수를 통해 시간이 단위에 맞게 증가할 수 있도록 수정하였다.

4. Global Time 모드에서 왼쪽 아래 버튼을 짧게 눌러 Global Time을 15분 씩 감소시킨다.

- Problem

- 시간을 감소시킬 수는 있으나 '15분 -> 30분 -> 45분'이 아닌 '15분 -> 45분 -> 1시간 30분'으로 감소함. 즉 더해지는 폭이 15분 씩 감소함

- Fixed

```
public GlobalTime() {
    this.gTime = super.time;
    this.modifiedMinute = 15;
    this.modifiedHour = 1;
    this.simpleDateFormat = new SimpleDateFormat( pattern: "yy-MM-dd HH:mm:ss");
    this.stringGTime = this.simpleDateFormat.format(this.gTime.getTime());
    this.timer.scheduleAtFixedRate(this.task,  delay: 0L,  period: 1000L);
}
```

```
public void saveMinutesMinus() { this.gTime.add( field: 12, -this.modifiedMinute); }
```

- modifiedMinue 변수를 통해 분이 단위에 맞게 감소하도록 수정하였다.

5. Global time 모드에서 왼쪽 아래 버튼을 길게 눌러 Global Time을 1시간씩 계속 감소시킨다.

- Problem

- 시간을 감소시킬 수 있으나 '1시간 -> 2시간 -> 3시간'이 아닌 '1시간 -> 3시간 -> 6시간'으로 감소함. 즉 더해지는 폭이 1시간씩 감소함.

- Fixed

```
public GlobalTime() {
    this.gTime = super.time;
    this.modifiedMinute = 15;
    this.modifiedHour = 1;
    this.simpleDateFormat = new SimpleDateFormat( pattern: "yy-MM-dd HH:mm:ss");
    this.stringGTime = this.simpleDateFormat.format(this.gTime.getTime());
    this.timer.scheduleAtFixedRate(this.task,  delay: 0L,  period: 1000L);
}
```

```
public void saveHoursMinus() { this.gTime.add( field: 10, -this.modifiedHour); }
```

- modifiedHour 변수를 통해 시간이 단위에 맞게 감소하도록 수정하였다.

6. Set mode 모드에서 오른쪽 위 버튼을 짧게 눌러 4가지 모드 중 현재 선택된 모드를 리셋한다.

- Problem

- 리셋이 되긴 하나 가끔씩 프로그램이 멈춰버린다. 안정성 문제 발생

- Fixed

```
public void exitSettingModes() {
    if (this.numberOfModes == 3) {
        System.arraycopy(this.newModes,  srcPos: 1, this.currentModes,  destPos: 1,  length: 5);
    }

    this.resetNewModes();
    this.modesArrayIndex = 0;
    this.isEnterSettingModes = false;
}
```

```
public void resetNewModes() {
    for(int i = 1; i < 6; ++i) {
        this.newModes[i] = false;
    }

    this.numberOfModes = 0;
}
```

- isEnterSettingModes 변수를 통해 리셋이 올바르게 되었는지 확인해준다.
- resetNewModes()를 통해 Set Mode 모드의 변수 안정성을 향상시켜 주었다.

7. timer 모드에서 시간을 00:00:00으로 세팅하고 시작 버튼을 눌러 timer를 실행시킨다.

- Problem

- 프로그램이 동작하지 않음

- Fixed

```
private float time = 0.0F;
private boolean isTimerStart = false;
private boolean isCountDownZero = false;
private boolean setBuzzerRing = false;
```

- 변수 초기값을 제대로 전달하여 프로그램이 동작하도록 수정하였다.

8. stopwatch 모드에서 시간이 00:00:00일 때 리셋 버튼을 눌러 리셋 후, 시작 버튼을 눌러 stopwatch를 실행한다.

- Problem

- 프로그램이 동작하지 않음.

- Fixed

```
public void resetStopwatch() {
    this.startStopwatch = false;
    this.stopwatchTime = 0L;
}
```

- resetStopwatch()로 분리하여 리셋이 올바르게 적용되고, 프로그램이 동작하도록 수정하였다.

9. stopwatch 모드에서 59분 59초에서 1시간 00분 00초로 넘어가는지 확인한다.

- Problem

- 59분에서 1시간으로 넘어가지 않음

- Fixed

```
TimerTask task = run() → {
        if (Stopwatch.this.startStopwatch && Stopwatch.this.stopwatchTime < 720000L) {
            ++Stopwatch.this.stopwatchTime;
        }

};
```

- 범위를 올바르게 설정해 줘서 시간단위가 넘어갈 수 있도록 수정.
- 최대 2시간 까지 표시되며 02:00:00에서 멈춤

10. 동일한 alarm 2개 이상 중복하여 설정 한 후, 설정한 시간이 되었을 때 alarm이 제대로 울리는지 확인한다.

- Problem

- 리셋이 되긴 하나 가끔 씩 프로그램이 멈춰버린다. 안정성 문제 발생

- Fixed

```
public boolean isAlarmActivated() {
    return this.alarm0.getIsAlarmActivated() || this.alarm1.getIsAlarmActivated() || this.alarm2.getIsAlarmActivated() || this.alarm3.getIsAlarmActivated();
}

public boolean isThisAlarmActivated() {
    switch(this.alarmIndex) {
    case 0:
        return this.alarm0.getIsAlarmActivated();
    case 1:
        return this.alarm1.getIsAlarmActivated();
    case 2:
        return this.alarm2.getIsAlarmActivated();
    case 3:
        return this.alarm3.getIsAlarmActivated();
    default:
        return false;
    }
}
```

- switch문이 중첩되지 않도록 하여 안정성을 향상시켰다.

11. alarm을 1개 설정한 후에, time 모드 화면으로 전환되고 기다렸다가 설정한 시간에 alarm이 제대로 울리는지 확인한다.

- Problem

- alarm이 제 시간에 제대로 울리지 않고, 설정한 시간에서 6초 정도 시간이 더 흐른 후에 울림.

- Fixed

- exception in thread "AWT-EventQueue-0" java.lang.NullPointerException 에러 수정 완료 -> 알람 제시간에 울린다.

12. alarm을 1개 설정한 후에, timer 모드 화면으로 전환되고 기다렸다가 설정한 시간에 alarm이 제대로 울리는지 확인한다.
- Problem
- alarm이 제 시간에 제대로 울리지 않고, 설정한 시간에서 6초 정도 시간이 더 흐른 후에 울림.
- Fixed
- exception in thread "AWT-EventQueue-0" java.lang.NullPointerException 에러 수정 완료 -> 알람 제시간에 울린다.

13. alarm을 1개 설정한 후에, stopwatch 모드 화면으로 전환되고 기다렸다가 설정한 시간에 alarm이 제대로 울리는지 확인한다.
- Problem
- alarm이 제 시간에 제대로 울리지 않고, 설정한 시간에서 6초 정도 시간이 더 흐른 후에 울림.
- Fixed
- exception in thread "AWT-EventQueue-0" java.lang.NullPointerException 에러 수정 완료 -> 알람 제시간에 울린다.

14. alarm을 1개 설정한 후에, global time 모드 화면으로 전환되고 기다렸다가 설정한 시간에 alarm이 제대로 울리는지 확인한다.
- Problem
- alarm이 제 시간에 제대로 울리지 않고, 설정한 시간에서 6초 정도 시간이 더 흐른 후에 울림.
- Fixed
- exception in thread "AWT-EventQueue-0" java.lang.NullPointerException 에러 수정 완료 -> 알람 제시간에 울린다.

15. timer를 1개 설정한 후에, time 모드 화면으로 전환되고 기다렸다가 설정한 시간에 timer alarm이 제대로 울리는지 확인한다.
- Problem
- timer alarm이 설정한 시간이 만료되어도 울리지 않는다.
- Fixed
- 알람 울림 기능 추가.

16. timer를 1개 설정한 후에, timer 모드 화면으로 전환되고 기다렸다가 설정한 시간에 timer alarm이 제대로 울리는지 확인한다.
- Problem
- timer alarm이 설정한 시간이 만료되어도 울리지 않는다.
- Fixed
- 알람 울림 기능 추가.

17. timer를 1개 설정한 후에, stopwatch 모드 화면으로 전환되고 기다렸다가 설정한 시간에 timer alarm이 제대로 울리는지 확인한다.
- Problem
- timer alarm이 설정한 시간이 만료되어도 울리지 않는다.
- Fixed
- 알람 울림 기능 추가.

18. timer를 1개 설정한 후에, global time 모드 화면으로 전환되고 기다렸다가 설정한 시간에 timer alarm이 제대로 울리는지 확인한다.

- Problem

- timer alarm이 설정한 시간이 만료되어도 울리지 않는다.

- Fixed

- 알람 울림 기능 추가.

19. Global time을 15분, 1시간 간격으로 내가 원하는 임의의 시간으로 설정할 수 있는지 확인한다.

- Problem

- 15분, 1시간 간격으로 시간이 증가 및 감소하지 않아 내가 원하는 임의의 시간으로 설정할 수 없다.

- Fixed

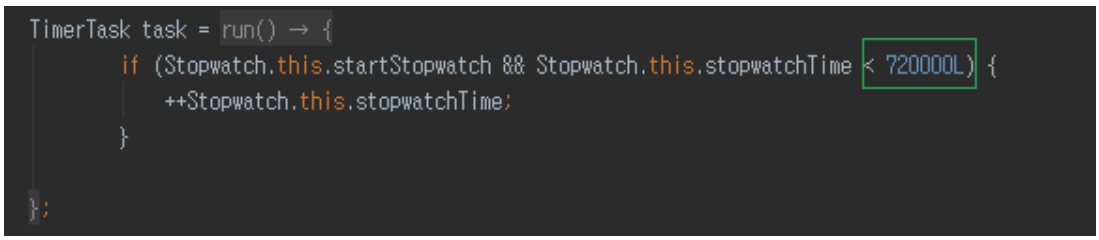
```
public GlobalTime() {
    this.gTime = super.time;
    this.modifiedMinute = 15;
    this.modifiedHour = 1;
    this.simpleDateFormat = new SimpleDateFormat( pattern: "yy-MM-dd HH:mm:ss");
    this.stringGTime = this.simpleDateFormat.format(this.gTime.getTime());
    this.timer.scheduleAtFixedRate(this.task,  delay: 0L,  period: 1000L);
}
```

```
public void saveHoursAdd() { this.gTime.add( field: 10, this.modifiedHour); }

public void saveHoursMinus() { this.gTime.add( field: 10, -this.modifiedHour); }

public void saveMinutesAdd() { this.gTime.add( field: 12, this.modifiedMinute); }

public void saveMinutesMinus() { this.gTime.add( field: 12, -this.modifiedMinute); }
```

- 15분, 1시간 간격으로 시간이 증가 및 감소할 수 있도록 수정해주었다.

20. set time 모드에서 1초, 1분, 1시간, 1일, 1달, 1년 단위로 내가 원하는 임의의 시간으로 설정할 수 있는지 확인한다.
- Problem
- 시간을 설정할 수 는 있으나 설정 값이 제대로 저장되지 않는다.
- Fixed
- 문제 없음

21. stopwatch의 최대 시간이 spec의 1시간 59분 59초와 부합하는지 확인한다.
- Problem
- stopwatch의 최대 시간을 spec의 1시간 59분 59초를 초과하여 2시간 이후도 가능함.
- Fixed



- 2시간까지 가능하며, 조건문을 사용해 범위에 들어왔을 때만 시간이 올바르게 흐르도록 수정하였다.

Activity 3. Static Analysis Response

1. bug

- Problem

```
[TravelerDigitalWatchJDKError/src/Alarm.java]
    1) Either re-interrupt this method or rethrow the "InterruptedException". Bug 해결 요망.
        → thread에게 interrupt를 발생시켜 주어야 함.
```

- Fixed

```
try {
    Thread.sleep( millis: 9);
} catch (InterruptedException e) {
    String exception= e.getMessage();
    System.out.println("UI Interrupted Exception occured : "+exception);
    Thread.currentThread().interrupt();
}
```

- thread에게 interrupt를 발생시켜주었다.

2. Vulnerability

- Problem

```
[TravelerDigitalWatchJDKError/src/NumberGenerator.java]
    1) The use of java.lang.Math.random() is predictable. 해결 요망.
      → 난수 생성 시 그냥 random() method 사용 시 예측이 가능할 수 있으므로 secureRandom 사용을 권
```

- Fixed

```
SecureRandom random;

try{
    random = SecureRandom.getInstance("SHA1PRNG");
}catch(NoSuchAlgorithmException e){
    String exception = e.getMessage();
    System.out.println("Random Number Generator Exception occurred : "+exception);
}
```

- 수정 완료

```
610 610                         gui.setIconInvisible(1);
611 611                         gui.modeDisplay(modeIndex);
612 612 +
613     -                       if((int)system.getTimerTime() != 0) {
    613 +
614 614                           hourPart = (int) (system.getTimerTime() / (60 * 60));
615 615                           minutePart = (((int) system.getTimerTime()) % 3600) / 60;
616 616                           secondPart = (int) (system.getTimerTime() % 60);
617     -                       } else {
618     -                           secondPart = 0;
619     -                       }
620 617
621     -                       System.out.println(secondPart);
    618 +                       //System.out.println(secondPart);
622 619                         gui.updateHour(hourPart, true);
623 620                         gui.updateMinute(minutePart, true);
624 621                         gui.updateSecond(secondPart);
```

펼쳐

```
1  + import javax.sound.sampled.AudioInputStream;
2  + import javax.sound.sampled.AudioSystem;
3  + import javax.sound.sampled.Clip;
4  + import java.io.File;
5  +
6  + public class Bell extends Thread {
7  +     private AudioInputStream audioIn;
8  +     private Clip clip;
9  +
10 +     public Bell() {
11 +         File f = new File("./Alarm.wav");
12 +         try {
13 +             audioIn = AudioSystem.getAudioInputStream(f.toURI().toURL());
14 +             clip = AudioSystem.getClip();
15 +             clip.open(audioIn);
16 +             clip.stop();
17 +         } catch (Exception e){
18 +             e.printStackTrace();
19 +         }
20 +     }
21 +     public void run(){
22 +         clip.loop(Clip.LOOP_CONTINUOUSLY);
23 +         clip.start();
24 +     }
25 +
26 +     public void play(){
27 +         this.start();
28 +     }
29 +
30 +     public void pause(){
31 +         clip.stop();
32 +     }
33 +
34 +     public boolean isPlaying(){
35 +         return clip.isActive();
36 +     }
37 + }⊘↵
```

```
632     -                       if(system.getStopwatchTime()<360000) {
633     -                           secondPart = (int) (system.getStopwatchTime());
634     -                           minutePart = (int) ((system.getStopwatchTime() / 100) % 60);
635     -                           hourPart = (int) ((system.getStopwatchTime() / 100) / 60);
636     -                       } else {
637     -                           secondPart = (int) ((system.getStopwatchTime() / 100) % 60);
638     -                           minutePart = (int) ((system.getStopwatchTime() / 100) / 60)%60;
639     -                           hourPart = (int) ((system.getStopwatchTime() / 100) / 3600);
640     -                       }
        634 +                   secondPart = (int) (system.getStopwatchTime());
        635 +                   minutePart = (int) ((system.getStopwatchTime() / 100) % 60);
        636 +                   hourPart = (int) ((system.getStopwatchTime() / 100) / 60);
641     637                     gui.updateHour(hourPart, true);
        ...
30      -                   System.out.println("Alarm Interrupted Exception occurred"+exception);
        31 +                Thread.currentThread().interrupt();
        32 +                System.out.println("Alarm Interrupted Exception occurred"+exception);
        ...
52      56      public void isAlarmSound(int currentHour, int currentMinute){     // 알람이 세 시간이 되어 울리게 되는지 알려주는 메소드.
53      -           if(this.alarmTimeHour == currentHour && this.alarmTimeMinute == currentMinute){
        57 +        if(this.alarmTimeHour == currentHour && this.alarmTimeMinute == currentMinute && this.alarmTimeSecond == 0){
54      58              buzzerOn = true;
```

Activity 4. OOPT Review

Review
- 후기
- 소프트웨어 모델링 실습을 진행하면서 소프트웨어를 단계적으로 개발하는 것이 얼마나 도움이 되는지 알 수 있었음 - 사용자의 입장에서 어떻게 소프트웨어를 구성하면 좋을지 고민하는 것부터 시작하여 고객이 원하는 소프트웨어를 개발하기 위해 단계적으로, 반복적으로 작업을 진행하면서 객체 지향 설계가 중요하다는 것을 깨달을 수 있었음. - OOAD 방법론을 통해 Stage를 진행하다 보면 몇몇 작업을 반복하는 느낌이 들어 다소 지루할 수 있지만, 실제 코드 구현을 시작할 때 클래스 간의 관계를 한 눈에 파악할 수 있어 쉽게 진행할 수 있었음 - 소프트웨어 검증을 통해 우리의 프로젝트의 결함을 파악하고 수정하는 단계를 거치면서 완벽한 소프트웨어를 만드는 것이 얼마나 많은 인력과 시간을 필요로 하는 일인지 실감할 수 있었음
- 소검팀과의 협업
- 소모팀에서 찾아내지 못하는 오류 발견에 큰 도움이 됨 - CTIP 환경 구축단계 어려움 -> 소검팀의 빠른 피드백이 있었음