

# Airtest IDE를 활용한 모바일 게임 테스트 자동화

## NO RISK, NO TEST

2019.11.28(목) 19:00~  
AWSKRUG #architecture 28번째 모임

[Two Hands Games] 글로벌사업부  
[Quality Assurance Engineer] 이지원

# INDEX

## 01

### 소개

01-1. 지나온 발자취

## 02

### 자동화 테스트의 필요성

02-1. Risk Based Testing 관점

02-2. Positive Testing과 경험기반 테스트의 관점

02-3. 결론

## 03

### Airtest 실무 활용 과정

03-1. Airtest 소개

03-2. 활용 과정

03-3. 트러블 슈팅

## 04

### Airtest 실무 활용 영상

04-1. Unity 기반 모바일 게임

## 05

### 향후 계획

05-1. QA 오프라인 소모임 개설

# 소개

## 01 지나온 발자취

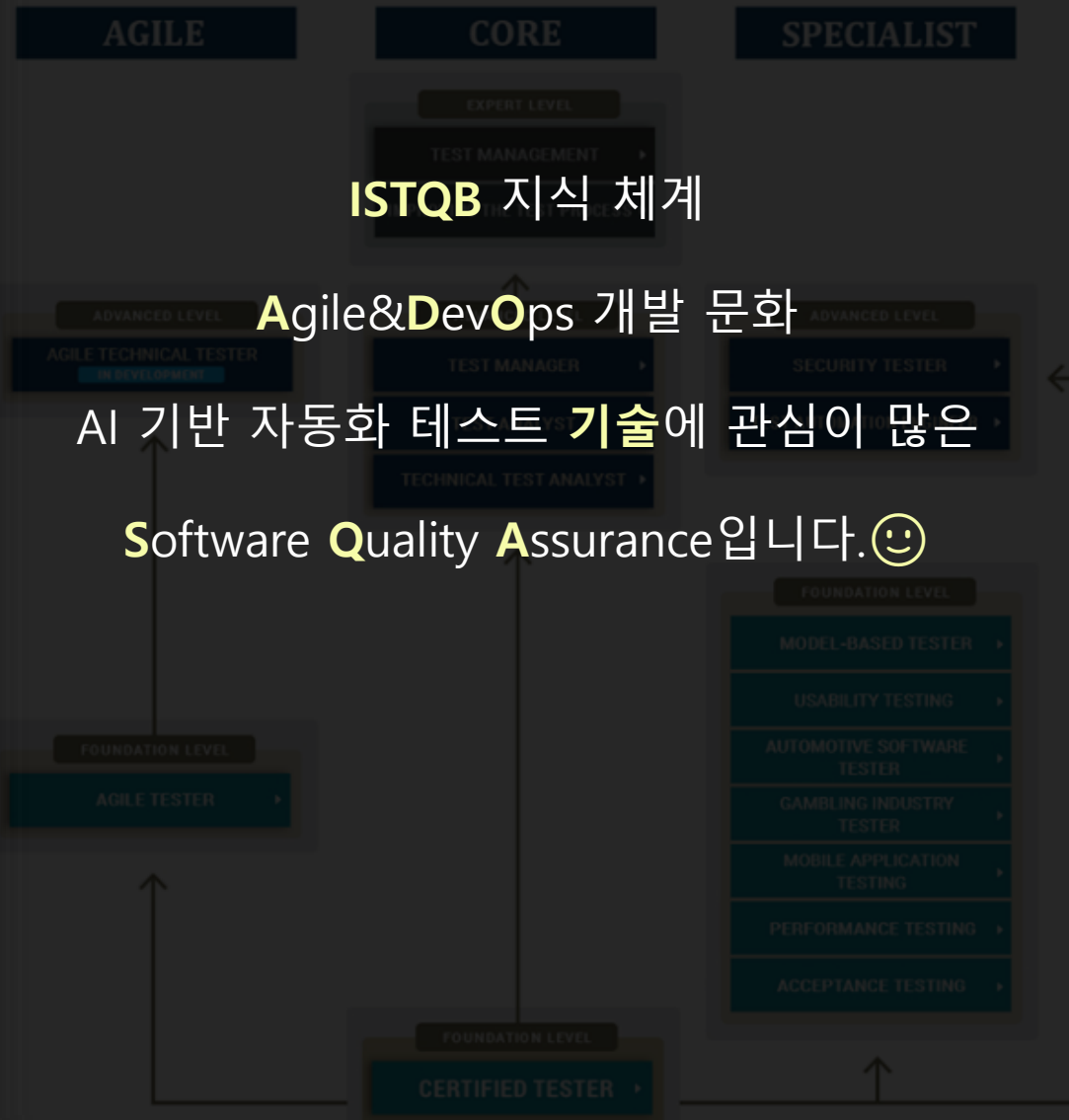
```
30 medalrace_go_my_position(template(r"tpl1573530263948.png", record_pos=(2.118, 0.646), resolution=(953, 1030)),
31 medalrace_my_grade_label(template(r"tpl1573530263948.png", record_pos=(2.207, 0.135), resolution=(953, 1030)),
32 medalrace_my_grade_label(template(r"tpl1573530263948.png", record_pos=(2.209, 0.112), resolution=(953, 1030)),
33 medalrace_my_grade_label(template(r"tpl1573530263948.png", record_pos=(2.207, 0.031), resolution=(953, 1030)),
34 medalrace_my_grade_label(template(r"tpl1573530263948.png", record_pos=(2.454, -0.377), resolution=(953, 1030)))
35
36 * assert_exists(medalrace_start_screen[0]):
37 * touch(medalrace_start_screen[0])
38 * assert_exists(medalrace_start_screen[1]):
39 * touch(medalrace_start_screen[1])
40 * assert_exists(medalrace_start_screen[2]):
41 * touch(medalrace_start_screen[2])
42 * else:
43 * touch(medalrace_start_screen[1])
44 * touch(medalrace_start_screen[0])
45 * if assert_exists(medalrace_start_screen[2]):
46 * touch(medalrace_start_screen[2])
47 * * 메달레이스 진입 완료후
48
49
50 * if assert_exists(medalrace_top_fixed_text):
51 * sleep(2)
52 * else:
53 * wait(medalrace_top_fixed_text)
54 * * 메달레이스 화면 정상 진입 확인
55
56
57 assert_not_exists(medalrace_go_my_position)
58 * * 내 메달 위치로 이동 버튼. 사전 조건은 메달레이스로 노출되면 진행
59
60
```

```
61 * touch(medalrace_reward_chest[0]):
62 * touch(medalrace_reward_chest[0])
63 * if assert_exists(medalrace_reward_singlecard[0]):
64 * touch(medalrace_reward_singlecard[0])
65 * else:
66 * touch(medalrace_reward_currency[0])
67 * if assert_exists(medalrace_reward_singlecard[0]):
68 * for i in range(1, 5):
69 * touch(medalrace_reward_singlecard[0])
70 * if assert_exists(medalrace_reward_currency[0]):
71 * break
72 * * 보상 끌어서부터 5회 터치)그와 예외 조건 케이스.
```

```
IDE_2019-09-10_py3_win64\airtestIDE_2019-09-11_py3_win64\airtest\core\android\static\adb\windows\adb.exe -P 5037 -s R3CM407SYZ shell getpr
IDE_2019-09-10_py3_win64\airtestIDE_2019-09-11_py3_win64\airtest\core\android\static\adb\windows\adb.exe -P 5037 -s R3CM407SYZ shell ls /
IDE_2019-09-10_py3_win64\airtestIDE_2019-09-11_py3_win64\airtest\core\android\static\adb\windows\adb.exe -P 5037 -s R3CM407SYZ shell ls /
IDE_2019-09-10_py3_win64\airtestIDE_2019-09-11_py3_win64\airtest\core\android\static\adb\windows\adb.exe -P 5037 -s R3CM407SYZ shell LD
2019-09-10_py3_win64\airtestIDE_2019-09-11_py3_win64\airtest\core\android\static\adb\windows\adb.exe -P 5037 -s R3CM407SYZ shell
```

# Career of the LEE JI WON

## PERSONAL HISTORY



# Career of the LEE JI WON

## PERSONAL HISTORY

*2016* → 2016.01 IGS 파견직

*2017* → 2017.01 IGS 정규직

*2018* → 2018.05 오드원게임즈 입사

*2019* → 2018.09 ~ 현재 투한즈게임즈

# Career of the LEE JI WON PERSONAL HISTORY

프로필&배너 제작을 도와주셨던  
마케팅 디자이너 최우인님 감사합니다.



2017

2017.03 Jio QA Life 페이스북 페이지 개설  
<https://www.facebook.com/ljwqalife/>

2019

2019.05 네이버 블로그 개설  
<https://blog.naver.com/wldnjs3027>





# Career of the LEE JI WON PERSONAL HISTORY

<https://www.sten.or.kr/index.php>



테스팅이야기 테스트 분야 실무, 노하우, 의견들을 공유하는 곳입니다.

알림가능

RSS

글쓰기

번호	제목	글쓴이	날짜	조회
5	QA와의 협업을 중요시하게 여기는 역량 있는 시스템 기획자... (4)	Jio QA Life	19-09-01	993
4	게임 산업에서의 TMMI의 가치와 활용에 대해서 (2)	Jio QA Life	19-04-06	777
3	게임 개발사에서의 테스트 및 QA 경험 (6)	Jio QA Life	19-04-06	1850
2	게임QA(아웃소싱)로 커리어를 이제 막 쌓아가고자 하시는 ... (9)	Jio QA Life	18-04-27	4198
1	리스크 기반 테스트에 대한 몇 가지 의문사항에 대해서. (3)	Jio QA Life	18-04-26	1710

▲ ▼

목록

글쓰기

자유게시판 STEN 회원 여러분의 자유로운 이야기가 오가는 곳입니다.

알림가능

RSS

글쓰기

번호	제목	글쓴이	날짜	조회
7	서울벤처대학원대학교 테스트 분야 석박사 과정이 없어... (5)	Jio QA Life	19-09-14	2479
6	진로와 자기계발에 대한 고민. (5)	Jio QA Life	19-08-07	2504
5	TestLink 세팅 후 로그인을 하면 메뉴가 안나옵니다. 이... (4)	Jio QA Life	19-06-14	2126
4	강원대학교 산업대학원 테스트 석사과정 진행중이신분 ... (6)	Jio QA Life	19-01-20	3141
3	게임 QA 직군 오프라인 모임 주최 (10)	Jio QA Life	18-07-01	5858
2	CTAL - TM 한글로된 샘플 시험 문제는 아직 따로 없는거... (2)	Jio QA Life	18-06-20	6674
1	국제 표준을 활용한 긍정적인 QA활동 사례가 있을까요? (1)	하츠코이	18-02-24	5195

▲ ▼

목록

글쓰기

테스틀이야기  
게임 개발사에서의  
테스팅 및 QA 경험  
By Jio QA Life 님



# 자동화 테스트의 필요성

01 Risk Based Testing 관점

02 Positive Testing과 경험기반 테스트의 관점

03 결론

2



## 테스팅 원리와 리스크

### 테스팅의 7가지 기본 원리

테스팅은 결함이 존재함을 밝히는 활동이다.

결함 집중

살충제 패러독스(Pesticide paradox)

완벽한 테스트는 불가능하다.



**NO RISK, NO TEST**

오류-부재의 귀변

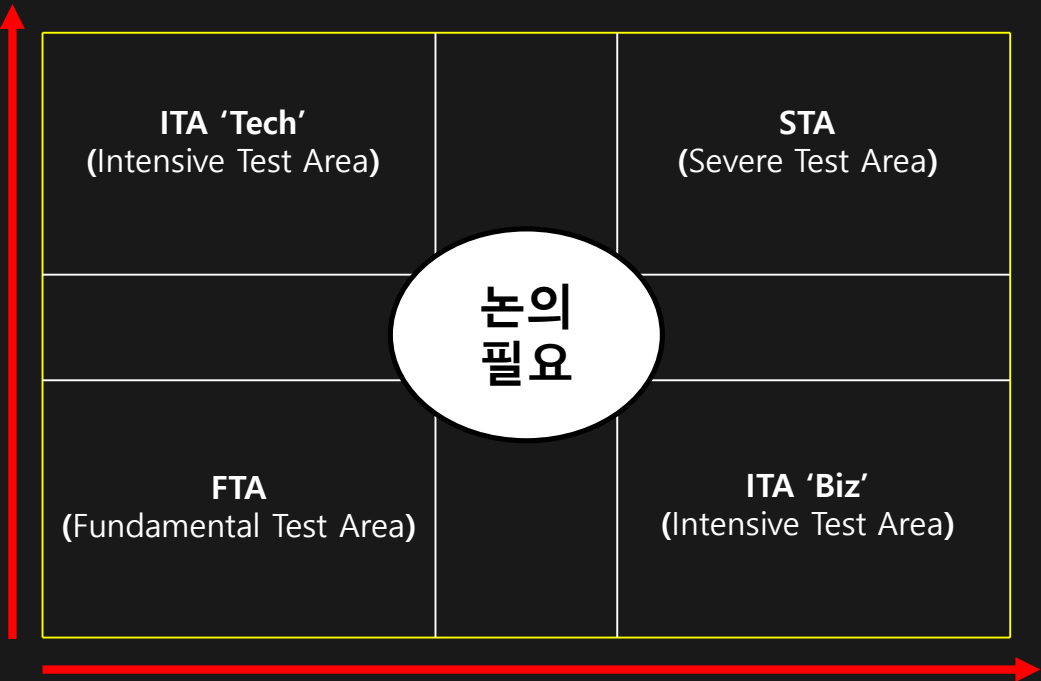
테스팅은 정황(Context)에 의존적이다.

테스팅을 개발 초기에 시작한다.

# Risk Based Testing 관점

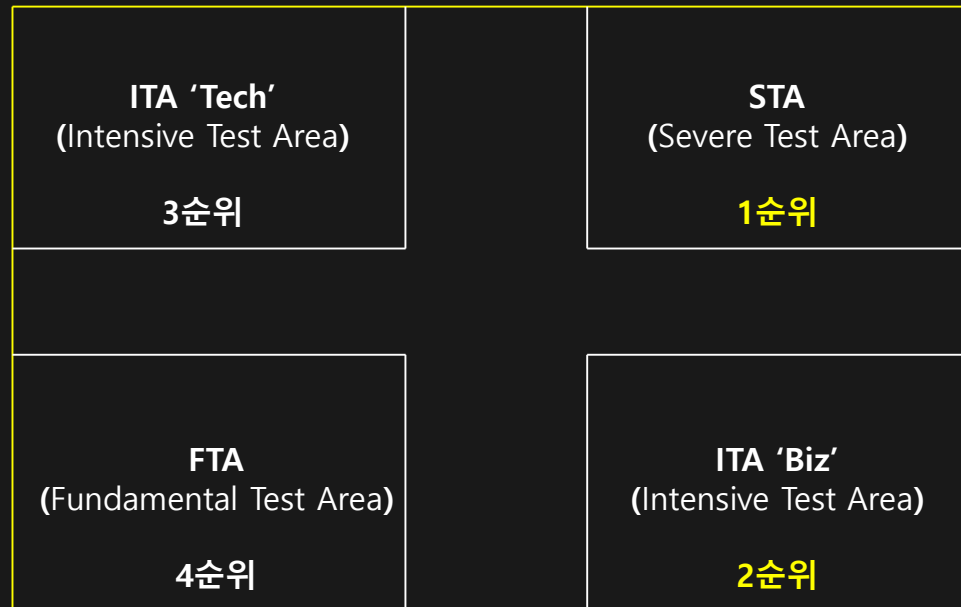
Risk Analysis Matrix

장애발생 가능성  
Likelihood  
(기술적 리스크)



장애로 인한 영향  
Impact  
(사업적 리스크)

## *High Level Test 우선순위*



# Risk Based Testing 관점

02-1

리스크 요소 리스크 아이템	장애 발생 가능성(Likelihood)					장애로 인한 영향도(Impact)				
	테스트 난이도	구현 난이도	상호작용	개발팀 성숙도	총합	사용자 중요도	운영 이슈	사용빈도	사용자 노출도	총합
메달 레이스 보상	1	3	3	0	7	9	9	5	5	28
도전 미션 보상	1	1	3	0	5	9	9	5	9	32
등급정보 UI	1	1	3	0	5	1	9	5	5	20
빅매치	9	5	5	0	19	5	9	9	9	32
클랜 레이스 연출	5	3	3	0	11	1	3	1	5	10
랭킹 UI	5	3	3	0	11	5	9	5	9	28
월드배틀 진행	1	3	5	0	9	9	9	9	9	36
클랜 창설	3	3	5	0	11	5	9	3	3	20
클랜 탈퇴	3	3	9	0	15	5	5	5	5	20
클랜원 강퇴	3	3	9	0	15	9	9	5	5	28
클랜 휘장	1	1	5	0	7	3	9	5	9	26
월드 배틀	1	1	2	0	4	3	8	2	8	28
월드 배틀 승리	3	3	8	0	14	8	8	2	2	38
월드 배틀 패배	3	3	8	0	14	2	2	2	2	20
월드 배틀 승리 보상	3	3	2	0	8	2	8	3	3	20

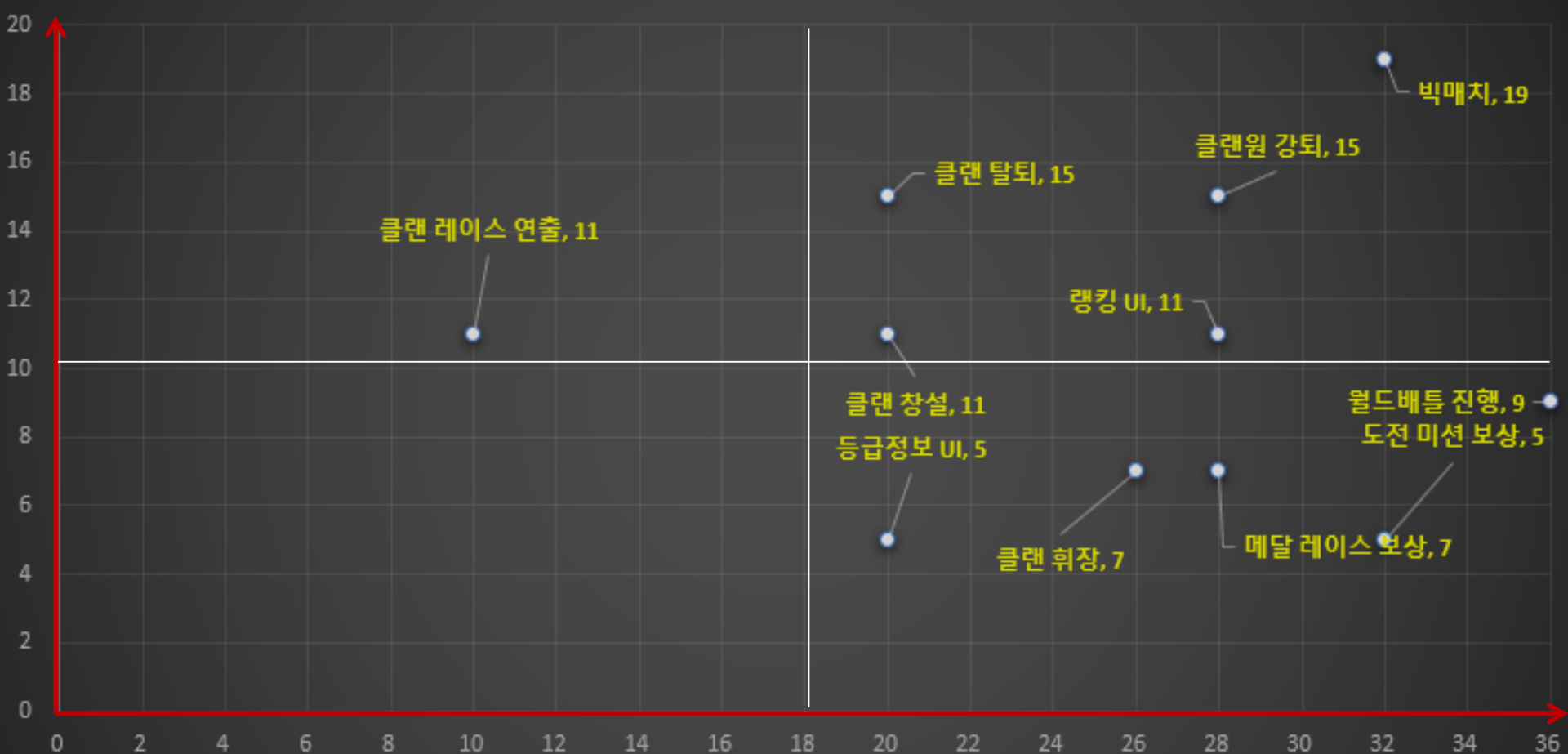
Impact 총합>18 AND 테스트 난이도, 구현 난이도, 상호작용의 값이 3이하인 아이템

ITa 'Biz'  
(Intensive Test Area)  
2순위

# Risk Based Testing 관점

Risk Scale	
9	심각(critical)
5	높음(high)
3	보통(moderate)
1	낮음(low)
0	없음(none)

리스크 분석 분포도 예시



# Risk Based Testing 관점

Risk Scale	
9	심각(critical)
5	높음(high)
3	보통(moderate)
1	낮음(low)
0	없음(none)

리스크 요소  리스크 아이템	장애 발생 가능성(Likelihood)					장애로 인한 영향도(Impact)				
	테스트 난이도	구현 난이도	상호작용	개발팀 성숙도	총합	사용자 중요도	운영 이슈	사용빈도	사용자 노출도	총합
메달 레이스 보상	1	3	3	0	7	9	9	5	5	28
도전 미션 보상	1	1	3	0	5	9	9	5	9	32
등급정보 UI	1	1	3	0	5	1	9	5	5	20
빅매치	9	5	5	0	19	5	9	9	9	32
클랜 레이스 연출	5	3	3	0	11	1	3	1	5	10
랭킹 UI	5	3	3	0	11	5	9	5	9	28
월드배틀 진행	1	3	5	0	9	9	9	9	9	36
클랜 창설	3	3	5	0	11	5	9	3	3	20
클랜 탈퇴	3	3	9	0	15	5	5	5	5	20
클랜원 강퇴	3	3	9	0	15	9	9	5	5	28
클랜 휘장	1	1	5	0	7	3	9	5	9	26
클랜 순위	1	1	2	0	4	3	9	2	9	28
클랜 순위	3	3	9	0	15	9	9	2	2	38
클랜 순위	3	3	9	0	15	2	2	2	2	20
클랜 순위	3	3	9	0	15	2	9	3	3	30

나에게 테스트 난이도 1의 의미는  
 ⇒ 단순 반복  
 ⇒ 지루함  
 ⇒ 내가 전문성이 있을까  
 ⇒ 이럴려고 공부하는게 아닌데



# Positive Testing과 경험기반 테스트의 관점

나는 자동화 테스트에 관심이 무척이나 많다.

Airtest라는 자동화 테스트 툴을 접하고 실무에 도입을 해보며 느낀 점은 잦은 UI 변경으로 인한 스크립트의 재사용성이 떨어진다는 점과 테스트 결과에 대한 신뢰성이 떨어진다는 점. 이뿐만 아니라 테스트 실행에 필요한 접근성이 낮다고 느껴졌으며 가장 중요한 것은 테스트 준비 기간에 직접 스크립트를 설계해야 하는데 이 자체로도 늘 일정과의 싸움이 빈번한 QA 팀 입장에서는 리소스가 소요되는 일이며 설계 방식 자체가 올바른 이미지를 찍어서 스크립트화 시키는 것이기 때문에 **Happy Path의 오류를 범하기 쉽다.**

General 한 방법으로 General 한 데이터를 입력하는 General 한 사용자가 과연 Major 하고 Critical 한 결함을 "잘" 찾아낼 수 있을까? Positive Case만 작성되어있는 Test case 1000줄이 있다고 할 경우 과연 해당 커버리지에서 결함이 얼마나 나올 것인가? 서비스에 치명적인 결함이 과연 발견될 수 있을 것인가? Test case 개수만을 테스트의 유일한 측정 매트릭스로 인식하고 수백수천 개의 Test case를 작성하고 유지 보수하며 매 수행 기간 때마다 **창의력이라곤 보이지 않은 기계처럼 수행하는 방식은 테스트는 창의적인 활동임에도 불구하고 이를 실천하지 않는 행위라 생각한다.**

특수한 이틀 뒤와와기 많은 해마다 해마다

매 수행 기간 때마다 창의력이든 아니든 많은 기계처럼 수행하는 방식은 테스트는 창의적인 활동임에도

이 창의력 부족 매트릭스로 인식하고 수백수천 개의 Test case를 작성하고 유지 보수하며

매 수행 기간 때마다 창의력이든 아니든 많은 기계처럼 수행하는 방식은 테스트는 창의적인 활동임에도 불구하고 이를 실천하지 않는 행위라 생각한다.

## 02-2 Positive Testing과 경험기반 테스트의 관점

서비스 출시 전 Major 하며 Critical 한 이슈를 이룬 시간에 찾아내어 품질을 컨트롤하기 위해서는 개인적으로 체계적인 절차에 의해 테스트 준비 단계부터 마감 활동이 이루어지는 Test Case 기반 테스트보단 James Bach가 주장한 Heuristic Testing인 Exploratory Testing을 프로세스화 시켜 조직 내에 해당 역량을

내재화 시키는 것이 적절하다고 생각한다.

Exploratory Testing은 기법이 아닌 방법론이기 때문에 조직 내에서 테스터의 지적 능력을 극대화할만한 테스트 환경과 협업 방식이 뒷받침되어야 효율성이 크고, 테스터의 역량에 따라 테스트 결과가 천차만별일 수도 있다는 점이 Risk로 다가오지만 이러한 Risk에 따른 반대 급부를 감재울만한 효율성이 분명 있으리라 생각한다.

조직의 상황에 따라, 프로젝트 성격에 따라 TC 기반 테스트와 경험 기반 테스트를 적절히 활용하는 것.  
그것이 Happy Path의 오류를 방지할 수 있는  
길이라 생각한다.

Airtest의 자동화 스크립트 설계와 수행 방식은 앞서 얘기한 Happy Path의 오류를 범하기 쉽고  
그러한 환경에 노출되어 있다.

---

이렇듯 제품의 User story가 필요한 상황이 아닐 경우엔  
Airtest의 테스트 결과는 크게 만족스럽지 못하였다.

이렇듯 Airtest의 테스트 결과는 크게 만족스럽지 못하였다.  
이렇듯 Airtest의 테스트 결과는 크게 만족스럽지 못하였다.

Positive 테스팅과 Negative 테스팅의 조화  
회귀 테스트의 효율성과 속도성 증가  
테스트 수행 누락&테스터의 컨디션 하락 예방  
단순 반복 업무를 벗어나므로 인해 생겨나는 긍정 효과  
애드혹&탐색적 테스팅의 효율성 극대화(테스터의 에너지 비축)

실무적인 관점도 중요하지만 가장 중요한 것은  
다양한 도메인에서 지속적인 연구와 효율성 입증으로  
**QA팀에서 주도한 자동화 테스트 도입의 성공이**  
**제품의 성공과 연결될 수만 있다면**

경영진의 인식 변화의 출발점이자  
기술 습득에 대한 QA 직군의 새로운 문화 형성이며  
QA 직군 인식 변화의 첫 단추가 될 것.

# Airtest 실무 활용 과정

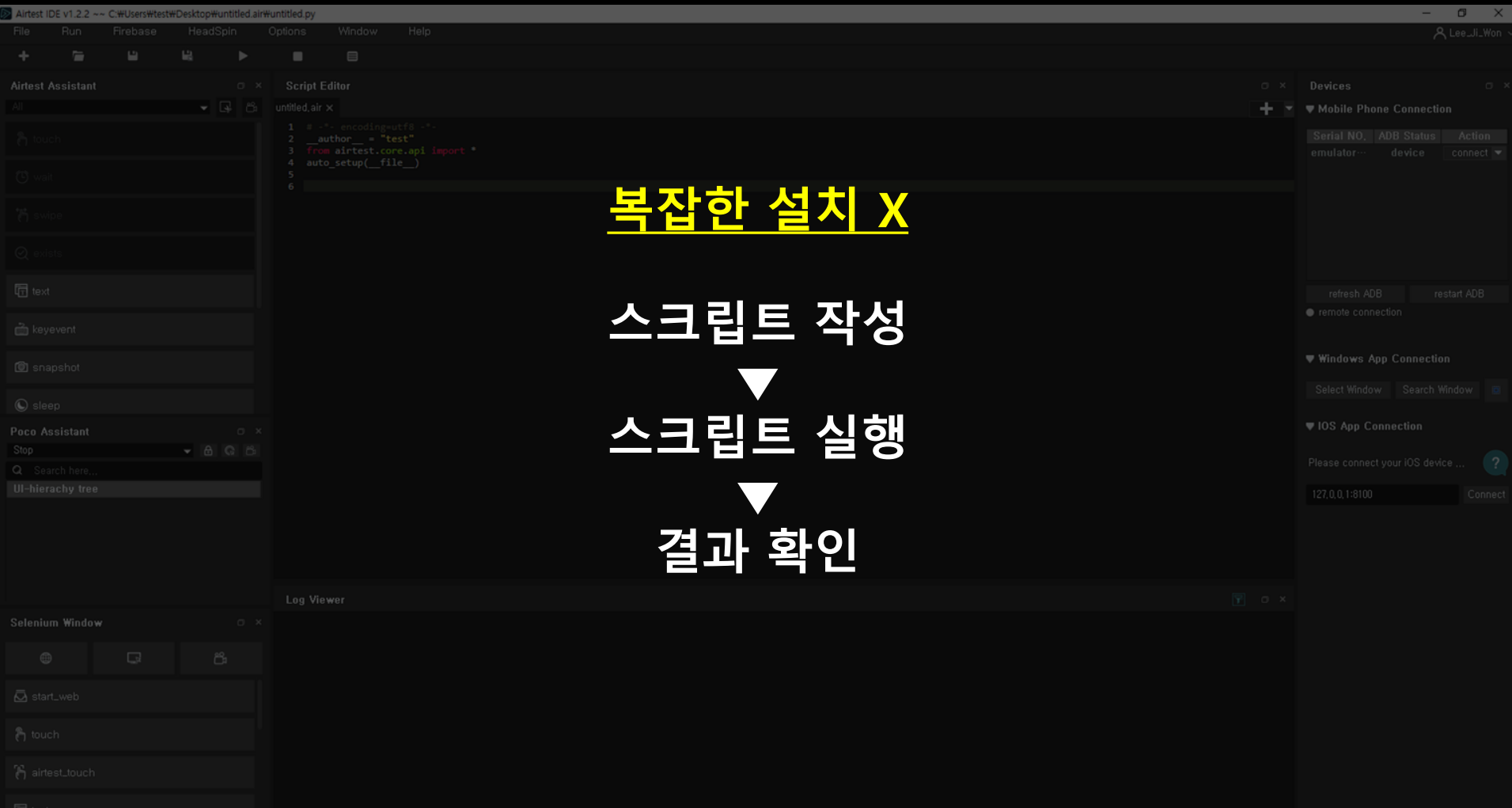
01 Airtest 소개

02 활용 과정

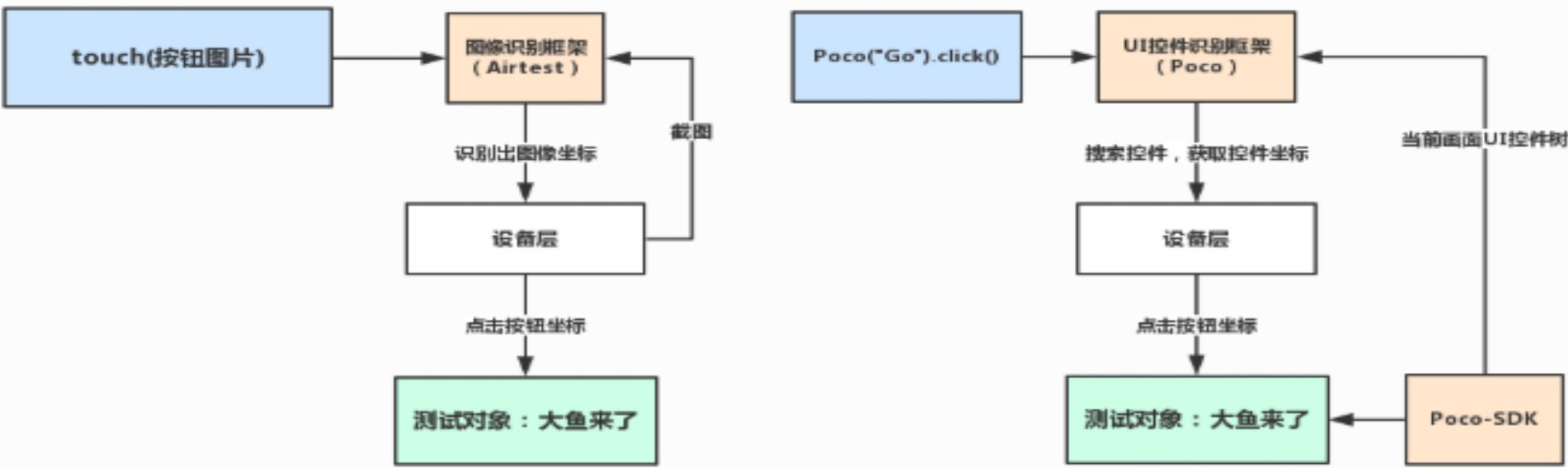
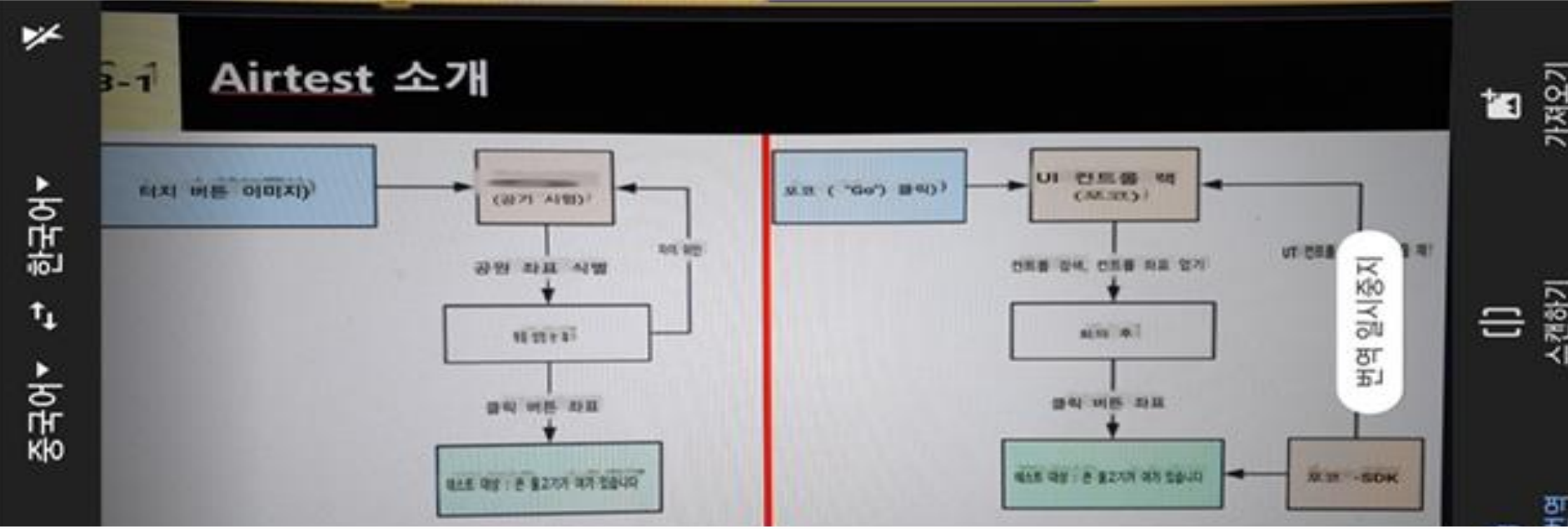
03 트러블 슈팅

3

# 03-1 Airtest 소개



# 03-1 Airtest 소개





# 03-1 Airtest 소개

Airtest IDE v1.2.2 -- C:\Users\test\Desktop\untitled.air\untitled.py

File Run Hirebase HeadSpin Options Window Help 5

**1**

**2**

**3**

**4**

**5**

Airtest Assistant

touch

wait

swipe

exists

text

keyevent

snapshot

sleep

Poco Assistant

Stop

Search here...

UI-hierarchy tree

Selenium Window

start\_web

touch

airtest\_touch

Script Editor

```
1 # -*- encoding=utf8 -*-
2 __author__ = "test"
3 from airtest.core.api import *
4 auto_setup(__file__)
5
6
```

Log Viewer

Devices

Mobile Phone Connection

Serial NO.	ADB Status	Action
emulator...	device	connect

refresh ADB restart ADB

remote connection

Windows App Connection

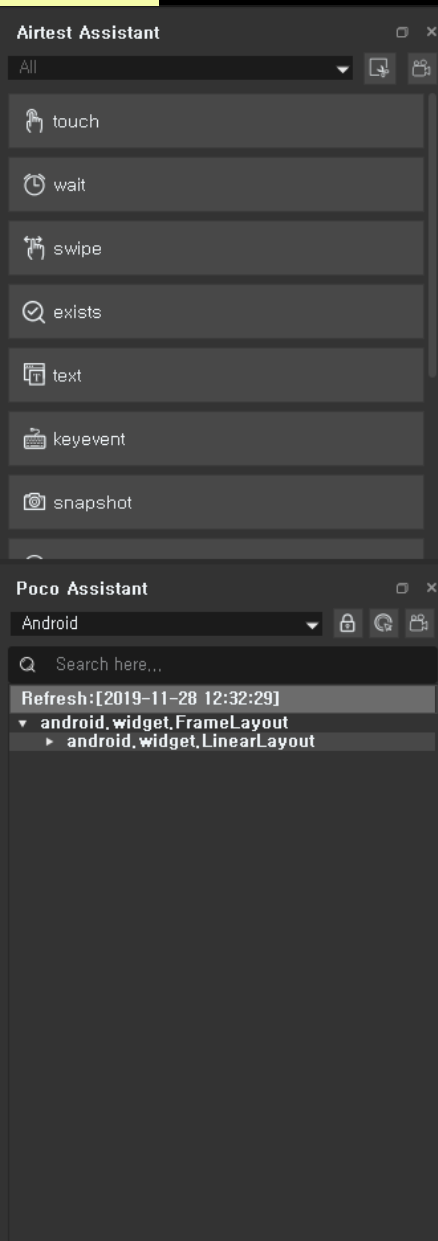
Select Window Search Window

IOS App Connection

Please connect your IOS device ...

127.0.0.1:8100 Connect

# 03-1 Airtest 소개



게임 및 앱을 위한 **이미지 인식 기술** 기반 UI 자동화 테스트 프레임 워크.  
Windows, Android 및 iOS 플랫폼을 지원함.

**터치&기다림&스와이프&존재확인&캡처&녹화&슬립&입력** 등 일반적으로  
테스트 케이스에서 실행 스텝 란에 작성되는 행위들을 명령문으로 제공함.

**UI 제어 인식**에 기반한 자동화 된 테스트 프레임 워크로  
현재 Unity3D / Android 기본 앱 / iOS 기본 앱 등을 지원함.

**게임 앱에 사용하려면 poco-sdk를 빌드에 구현 시켜야 함.(사용해볼 예정)**

## 2. 이론적 배경

### 2.1. SIFT 알고리즘

SIFT (Scale Invariant Feature Transform) 알고리즘은 2004년에 David Lowe 에 의해 처음 제안되었으며, 영상 회전, 스케일 변화, 유사성을 가진 변형(Affine Deformation), 관점 변화(Viewpoint Change), 잡음(Noise), 조명 변화(Illumination Change)에 매우 강인성을 가진 특징 추출 알고리즘이다[5]. SIFT는 크게 4 단계를 거쳐 수행되며, 이는 (1) 스케일 공간 극값 검출(Scale Space Extrema Detection), (2) 주요 점 지역화(Key Point Localization), (3) 방향성 배치(Orientation Assignment)와 (4) 기술 내용 생성(Description Generation)이다.

첫 번째 단계로, 서로 다른  $\sigma$ 값을 갖는 가우시안 차이(DoG, Difference-of-Gaussian) 함수에서 스케일 공간 극값을 사용하여 주요 점들의 위치와 스케일을 식별한다. DoG 함수는 상수 계수인  $k$ 로 분리되는 스케일 공간에서 영상을 컨볼루션 연산을 수행하며, 수식은 다음과 같다.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (1)$$

수식 (1)에서  $G$ 는 가우시안 함수이며,  $I$ 는 영상이다. DoG 를 산출하기 위해 가우시안 영상이 차감되고, 인자 2로 서브샘플링을 수행한다. 그런 다음, 해당 영상에서 DoG 를 산출한다.  $D(x, y, \sigma)$ 의 영역 최대값과 최소값을 검출하기 위해 하나의 픽셀에서 3\*3 인접 영역과 비교한다.

두 번째 단계에서, 낮은 대비 점들을 제거하여 주요 점들의 후보들을 지역적 집중화(Localization)하고 재정렬시킨다. 헤시안 행렬(Hessian Matrix)을 이용하여 주 곡률(Principal Curvature)을 계산하고, 곡면상의 공간 곡선이 갖는 최대와 최소의 곡률보다 큰 비율을 갖는 주요 점들을 제거한다.

세 번째 단계에서, 지역 이미지 기울기를 기반으로 주요 점의 방향을 구한다. 방향 설정을 얻기 위해, 주

Fig. 1. SIFT (Scale Invariant Feature Transform)

요 점이 위치한 주변 영역 내에서 샘플 점들의 기울기(Gradient) 방향으로부터 방향 히스토그램(Orientation Histogram)을 구한다.

마지막 네 번째 단계에서는, 주요 점을 중심으로 지역 내에 있는 각 영상 샘플 점에 대해 영상 기울기 크기와 방향을 기반으로 각 주요 점들에 대한 지역 영상 기술자(Local Image Descriptor)를 계산한다. 여기에서 언급된 샘플 점(Sample Point)은 기울기의 위치와 방향에 대한 3D 히스토그램을 구성하는 샘플이며, 각각은 4\*4 배열 위치 그리드(Grid)와 8 방향 빈(Bin)을 가진다. 따라서 주요 점 기술자(Key Point Descriptor)는 128-차원 벡터 값이 된다. Fig. 1은 주요 점 기술자에 대한 계산 방법을 설명한다.

먼저, 영상에 대한 가우시안 블러(Gaussian Blur) 레벨을 선택하기 위해 주요 점의 스케일(Scale)을 사용하여 주요 점 위치의 주변 기울기 크기와 방향을 구한다. 방향 불변성과 기술자의 좌표를 뽑기 위해, 주요 점 방향을 기준으로 기울기 방향을 회전시킨다. Fig. 1의 좌측에서 각 샘플 위치는 작은 화살표로 표시되었다. 우측은 주요 점 기술자를 표시한다. 4\*4 샘플 영역을 통해 축적된 방향 히스토그램을 생성함으로써, 기울기 위치에 확실한 변화가 발생할 수 있으며, 각 방향 히스토그램은 8개 방향성을 갖는다.

### 2.2. SURF 알고리즘

SURF (Speeded Up Robust Features) 알고리즘은 다중-스케일 공간 정리(Multi-Scale Space Theory)을 기반하며, 특징 기술자는 성능과 정확성에서 우수한 헤시안 행렬(Hessian Matrix)을 기반으로 검출된다. 행렬을 구하기 이전에, 계산량을 줄이기 위해 적분 영상(Integral Image)을 추출한다. 적분 영상은 Paul이 제안한 방법을 사용하며[7],  $x = (x, y)$  위치에서 적분 영상은 원래의 위치(Origin)와  $x$ 로 만들어지는 직사각형 영역 내에 존재하는 입력 영상  $I$ 에서의 모든 픽셀들의 합으로 표현되며, 이는 수식 (2)와 같다.

## 이미지 인식 알고리즘

```
from airtest.core.settings import Settings as ST
ST.CVSTRATEGY = ["tpl", "kaze", "brisk", "akaze", "orb", "sift", "surf", "brief"]
```

**장면, 이미지 해상도 및 플랫폼별로 각 알고리즘의 성능이 다르다고 함.**

**따라서 제품에 적합한 알고리즘을 찾아서 적용시키는 것이 필요해 보임.**

**레퍼런스를 찾고 있는데 배경지식 부족으로 인해 어려움을 겪고 있음.**

**어디서부터 어떻게 학습하면 좋을까요?**

## 03-2 활용 과정

Script Editor


```

94 touch(Template(r"tpl1559375490011.png",
95 record_pos=(0.278, 0.667), resolution=
96 (1440, 3040)))
97 sleep(0.5)#팝업을 다시 띄우고
98 touch(Template(r"tpl1559376204551.png",
99 record_pos=(0.252, 0.025), resolution=
100 (1440, 3040)))
101 sleep(0.5)#팝업을 닫는다
102 print("C-1. 1그레이드 등장 카드 13종 확인")
103 assert_exists(Template(r"tpl1559376372897
104 .png", record_pos=(0.001, -0.06),
105 resolution=(1440, 3040)))
106 sleep(0.5)
107 print("C-2. 좌측 최상단부터 우측으로 한줄씩
108 카드정보 모두 확인 후 마지막 카드정보 팝업창
109 닫기")
110 print("C-2. 캐논")
111 touch(Template(r"tpl1559376492387.png",
112 record_pos=(-0.253, -0.37), resolution=
113 (1440, 3040)))
114 sleep(0.5)#카드정보 팝업 띄우고
115 exists(Template(r"tpl1559378497186.png",
  
```

Log Viewer

```

[10:53:41][DEBUG]<airtest.core.api> try match
with SURFMatching
[10:53:42][DEBUG]<airtest.aircv.keypoint_base>
[SURF] threshold=0.7, result={'result': (354,
986), 'rectangle': [(228, 836), (228, 1137),
(480, 1137), (480, 836)], 'confidence':
0.980026513338089}
[10:53:42][DEBUG]<airtest.aircv.keypoint_base>
find_best_result() run time is 1.29 s.
[10:53:42][DEBUG]<airtest.core.api> match
result: {'result': (354, 986), 'rectangle':
[(228, 836), (228, 1137), (480, 1137), (480,
836)], 'confidence': 0.980026513338089}
[10:53:43][INFO]<airtest.core.api> Try finding:
  
```



[10:53:43][DEBUG]<airtest.core.api> try match with SURFMatching

Device Screen

이지원  
6월 1일

수백수천 개의 테스트 케이스 결과의 정확성은 수행자의 심리 상태나 당시의 컨디션과도 연관성이 있다고 생각합니다. 특히나 경험 기반 중에서 테스트의 설계, 수행, 계획, 테스트 기록 및 학습을 동시에 진행하는 휴리스틱 테스트 접근법인 탐색적 테스트만 봐도 컨디션이 정말 중요하다고 생각이 되는데 빌드가 5~6차례 쏟아져 나오면 직분을 알면서도 악마의 유혹이 가끔 찾아올 때가 있습니다.

테스터에게 유혹이 가장 쉽게 찾아오는 케이스들을 자동화 시켜버린다면 얼마나 더 생산적인 활동을 할 수 있을까라는 생각에서 시작된 게임 부분 자동화가 실무 도입을 앞두고 몇 차례 검증을 진행 중에 있는데 결과물이 꽤나 흥미롭네요.

다만 레퍼러스를 못 찾는 거지 정말 어느 거지. 참고하면 자료가 너무너무 없어서 660개 케이스를 자동화 시키는 스크립트를 작성하고 5차례 정도 검증하는 데에만 10시간이 걸렸네요.

예상치 못한 문제점들이 발생이 되어서 애를 좀 먹었는데 트러블 슈팅 간에 알게 된 팁이나 정보 같은 것들, 그리고 스크립트 작성 전에 사전 계획과 어떤 부분을 중점을 두고 진행되었는지, 실무에는 어떠한 변화가 일어났는지 등을 종합적으로 정리하여 제가 운영 중인 블로그에 꾸준히 올릴 예정입니다 😊

<https://blog.naver.com/wldnjs3027>

게임QA 부분 자동화에 관심 많으신 분들 같이 정보 공유하면서 소통해보요!

(첨부한 영상과 사진은 실제 스크립트 실행 과정 및 HTML로 뿌려지는 결과 화면입니다. 자체 학습만을 이용해도 어느 정도 덤스가 있는 기능 플로우 검증은 가능하며 A<>B 기능 간의 상호작용되는 부분도 파이썬을 이용하면 보다 손쉽게 작성이 가능해 보입니다. 단, 블로그에도 언급할 예정이지만 특정 상황에서는 100% 결과를 신뢰하기 어렵습니다.)

www.BANDICAM.com

GRADE  
Gateway City

3 Lv. 1 Cannon

Rarity	Type	Target	Number of Cards
Normal	Building	Ground	-

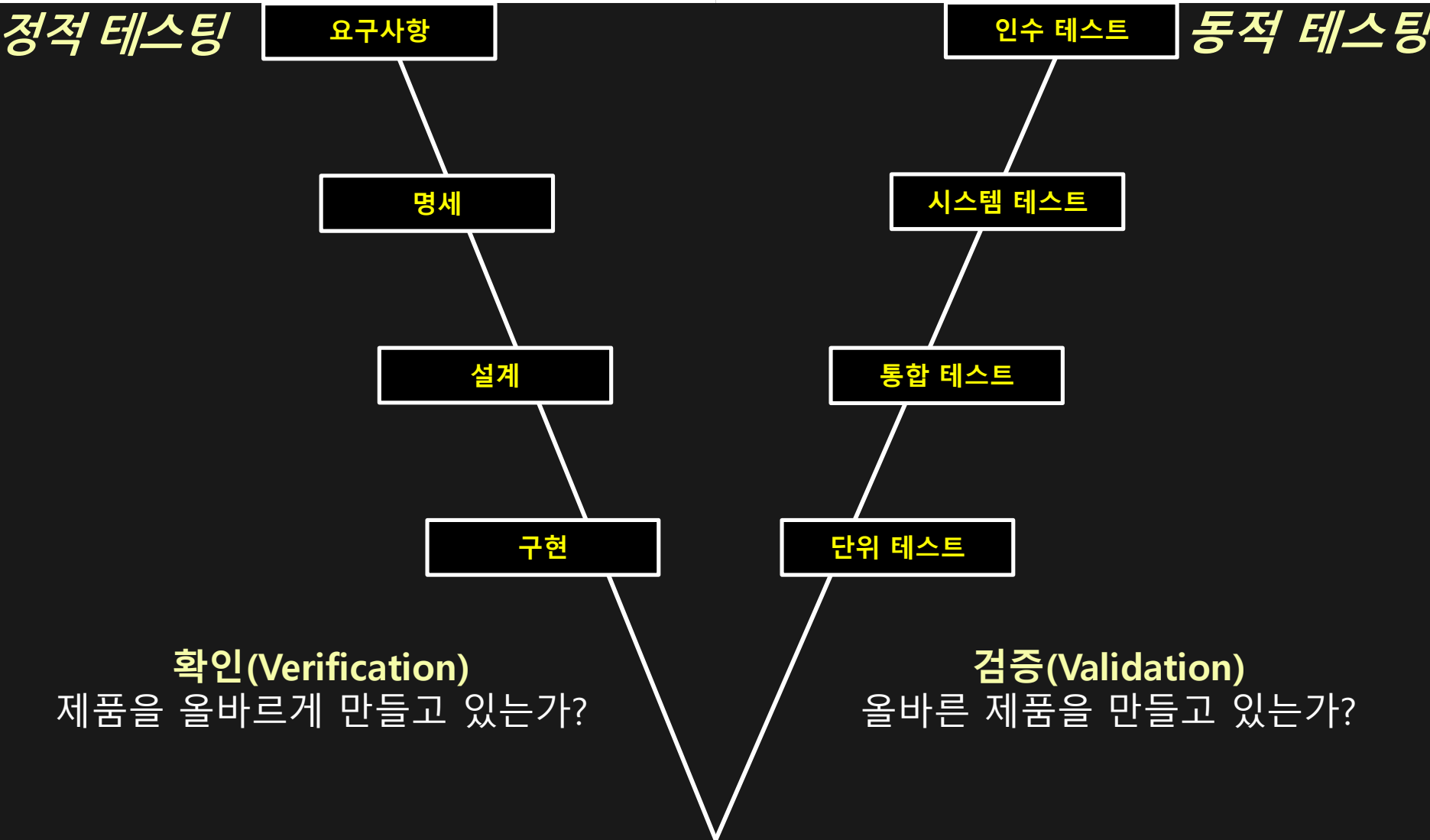
Hit Points 1057	Hit Speed 1.2 sec(s)
Damage 240	Duration 30 sec(s)
Damage/sec 199	Range 6
Deploy Time 1 sec(s)	

Confirm

1. 스크립트 작성 시간
2. 스크립트 실행 시간
3. 스크립트 재사용성 및 유지보수
4. 테스트 결과에 대한 신뢰
5. PASS 결과의 기준은?

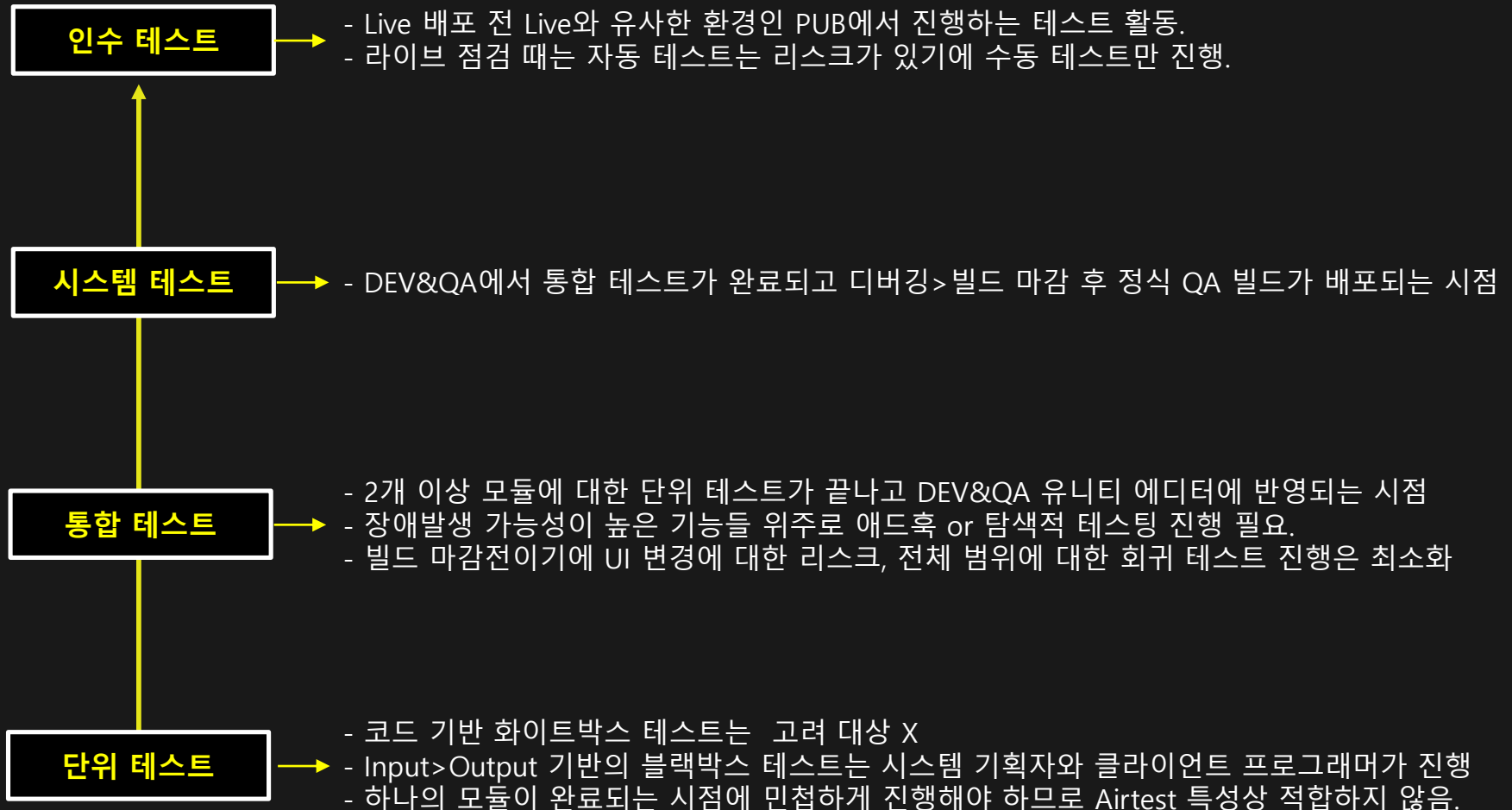
그냥 하지말까...

V모델로 바라본 테스트 프로세스  
*Verification and Validation*





## 테스트 레벨 정의 및 활용 레벨



## 테스트 대상

## &lt;점수를 획득하여 구간별 보상을 획득하는 콘텐츠&gt;

## 1. 테스트 중점 사항(CL)

- 구간 별 보상 획득이 가능하다.
- 보상 획득 가능 상태에 대한 UI 색상이 초록색으로 노출된다.
- 보상을 받고 나면 해당 UI 색상은 흑백으로 노출된다.
- 보상을 받고 나면 해당 UI에 체크 표시가 노출된다.
- 중복 보상 수령이 불가능하다.
- 도움말 텍스트 열기가 가능하다.
- 도움말 텍스트 닫기가 가능하다.

## 2. 추가 검증 가능 항목(CL)

- 그레이드 상승 시 상승 연출이 노출된다.
- 그레이드가 정해진 점수 구간대에 상승된다.
- 그레이드 상승 연출 후에 그레이드 정보 화면이 해당 그레이드로 표기된다.

## 3. 적용 이점

단순 반복적인 월드배틀 플레이를 통해 보상을 하나씩 받아 가며  
점수 상승->보상 획득의 시나리오를 유저처럼 테스트 가능

## 테스트 대상

## &lt;점수를 획득하여 구간별 보상을 획득하는 콘텐츠&gt;

## 4. 스크립트 작성 전 고려 항목(테스트 디바이스 사양 기준)

- 보상 획득 연출이 진행되는 시간은?
- 토스트 메시지가 사라지는데 걸리는 시간은?
- 화면에 보이는 보상 구간에 테스트 확인 항목과 동일한 이미지가 존재하는가?
- 콘텐츠 진입>홈 화면 나오기에 소요되는 시간은?
- 점수 획득으로 인한 그레이드 상승 시, 그레이드 상승 연출이 진행되는 시간은?
- 콘텐츠 진입에 필요한 UI가 가변적으로 바뀌는데 이미지를 어떻게 찾아낼 것인지?
- 배틀 종료>콘텐츠 진입하기 전에 점수 획득으로 인해 발생하는 연출을 넘기기 위한 예외 처리 방식은?

## 5. 예상 스크립트 소요시간

화면 전환, 연출에 대한 시간을 측정해야 하므로 현재까진 예상 불가

## 6. 테스트 결과 신뢰성

threshold (float) - 이미지 일치 임계 값을 디폴트 값 0.6에서 조금씩 상승>테스트 반복  
target\_pos (int) - 스크립트 실행 대상에 대한 이미지들을 인게임에서 체크하여  
이미지 일치 결과에 가장 적합한 위치로 설정  
RGB (bool) - 기본값은 False인데 사용할 경우 어떤 변화가 생기는지 반복 테스트 필요

## 03-3 트러블 슈팅

1. 테스트 케이스의 수행 완료 조건까지 실행되는 모든 구간마다 `assert_exists` 로직을 사용할 필요가 없다.
2. 사람의 판단이 필요한 기능이라면 수동 테스트를 하자.
3. Assistant에서 제공하는 단일 로직만을 사용하지 말고 Python 조건 반복문을 적극 활용하자.
4. 테스트 대상과 목적을 명확히 하자.
5. 동일한 스크립트를 실행해도 테스트 결과는 항상 동일하지 않기 때문에 충분한 사전 검증이 필요하다.
6. UI가 자주 변경되는 기능이라도 스크립트 유지보수에 큰 문제가 없다.
7. 이미지 찾기에 자주 실패한다면 1차적으로 `sleep` 값과 `wait` 값을 적절히 활용하자.
8. 텍스트보다 아이콘이나 버튼 같은 UI를 찾는 게 더 빨리 찾는다.
9. 이미지는 변수로 만들어두거나 리스트로 관리하자.
10. 담당 서비스에 대한 이해도가 높아야 한번 작성된 스크립트를 오래 사용할 수 있다.
11. 이미지 녹화 기능을 적극 활용하자.
12. 포기하지 말자.

4

유니티 기반 모바일 게임 활용 예시



# 04-1 Unity 기반 모바일 게임

## Airtest Report



Airtest Report

[Passed]

2019 / 11 / 27

23:05:07-23:24:51

Steps: 430

Time: 19min 43s 799ms

Executors

● Author: 이지원

● Airtest\_에시\_1128.air

order duration status

[Jump to wrong step](#)

Filter by:

All

Success

Failed

Assert

✓ # 1 Exists	1s 729ms
✓ # 2 Touch	3s 63ms
✓ # 3 Touch	1s 385ms
✓ # 4 Sleep	15s 0ms
✓ # 5 Touch	1s 836ms
✓ # 6 Touch	1s 953ms
✓ # 7 Touch	1s 946ms
✓ # 8 Touch	1s 906ms
✓ # 9 Sleep	3s 0ms
✓ # 10 Wait	8s 983ms
✓ # 11 Exists	1s 716ms
✓ # 12 Touch	1s 805ms

### Passed Step 1: Image not exists

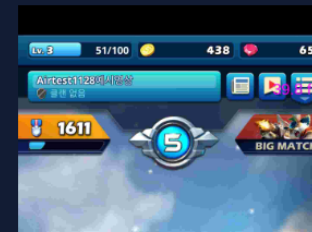
Status: Passed ✓

Duration: ⌚ 1s 729ms

Behavior: exists

Args:

resolution: 1440,3040



실행 개수&전체 실행 시간&개별 실행 시간&결과 필터링&테스트 포인트

# 04-1 Unity 기반 모바일 게임

order duration status

✓ # 421 Wait	1s 573ms
✓ # 422 Touch	1s 666ms
✓ # 423 Touch	1s 780ms
✓ # 424 Sleep	5s 0ms
✓ # 425 Exists	1s 236ms
✓ # 426 Touch	1s 439ms
✓ # 427 Exists	3s 449ms
✓ # 428 Touch	1s 847ms
✓ # 429 Exists	1s 792ms
✓ # 430 Touch	1s 967ms

< 1 - 20 21 22 > 20 22 Total 430

Jump to wrong step

Filter by: All Success Failed Assert

## Passed Step 429: Image exists

Status: Passed ✓

Duration: ⌚ 1s 792ms

Behavior: exists

### Args:

resolution: 1440,3040  
Confidence: 0.9720284640789032



실행 개수&전체 실행 시간&개별 실행 시간&결과 필터링&테스트 포인트

# 향후 계획

## 01 QA 오프라인 소모임 주최

5

# QA 소모임 개설

18년 7월 1일에 STEN 커뮤니티에 작성했던 QA 오프라인 모임 인원 모집 글 링크

18년 8월 10일에 참석 의사를 밝혀 주셨던 7분과 함께 첫 모임을 진행 할 계획이었지만  
개인적인 사정으로 진행하지 못하게 되었습니다.

다가올 2020년에는 보다 성숙해진 모습으로 다시한번 소모임을 주최해보려 합니다.

저와 함께 앞으로 소모임을 이끌어갈 운영진 분들도 모시고 있으며

관심 있으시다면 언제든지 연락 주세요 😊

이지원 / 카카오톡 ID: wldnjs3052

# Thank you

들어주셔서 감사합니다. 😊