The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 18, No. 2, pp.163-168, Apr. 30, 2018. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2018.18.2.163
JIIBC 2018-2-20

윈도우 환경에서의 GUI 기반 블랙박스 테스트 자동화 프로그램 도구

GUI-based Black Box Test Automation Program Tool in Windows Environment

정범진*, 이정우*, 홍창완*, 안병구**

Beomjin Jeong*, Jungwoo Lee*, Changwan Hong*, Beongku An**

요 약 본 논문에서는 윈도우 환경에서 블랙박스 테스트 기법을 사용하여 GUI 기반 테스트를 자동화하는 테스트 자동화 프로그램 도구를 제안 및 개발한다. 제안된 테스트 자동화 프로그램 도구의 주요한 특징은 다음과 같다. 첫째, 에러 상태를 이미지로써 지정하고, 테스트 스텝마다 화면을 캡처하여 이미지 유사도 비교를 통해 에러 메시지 검출 여부를 확인한다. 둘째, 실행 중 이벤트 대기시간이나 각 테스트 스텝 간 좌표 증가 값 등 여러 옵션 설정을 지원한다. 이러한 블랙박스 테스트 자동화 연구는 안드로이드나 웹 등의 환경에서는 많았지만 윈도우 환경에서는 그렇지 않았다. 제안된 시스템의 성능평가 결과 제안된 시스템은 이미지 비교 모듈로써 GUI 테스트 자동화를 수행하고, 프로세스 상태 확인과 에러 이미지 검출 여부를 확인함으로써 테스트를 정상적으로 수행함을 확인하였다.

Abstract In this paper, we propose and develop a test automation program tool that automates GUI based testing using black box testing technique in Windows environment. The main features of the proposed test automation program tool are as follows. First, an error condition is designated as an image, a screen is captured for each test step, and an error message is detected through comparison of image similarity. Second, the proposed system supports various setting options such as event waiting time during execution and coordinate increment value between each test step. Such black box test automation research was common in environments such as Android and Web, but not in Windows environment. The results of performance evaluation show that the proposed system performs GUI test automation as an image comparison module and confirms that the test is performed normally by confirming process status and error image detection.

Key Words : Software Testing, Black Box Test, Test Automation, Image Comparison

I. 서 론

소프트웨어 테스트란, 소프트웨어가 요구사항에 맞게 동작하여 입력에 대한 적절한 결과를 출력하는지 검사하는, 개발된 소프트웨어의 오류를 검출하는 과정이다. 이는 소프트웨어가 아무런 오류 없이 완전하다는 것을 증명하기 위한 것이 아니라 숨어있는 결함을 찾기 위한 과정이다. 특히 블랙박스 테스트는 프로그램의 구조를 고

*정회원, 홍익대학교 컴퓨터정보통신공학과
**종신회원, 홍익대학교 소프트웨어융합학과
접수일자: 2018년 2월 20일, 수정완료: 2018년 3월 20일
게재확정일자: 2018년 4월 6일

Received: 20 February, 2018 / Revised: 20 March, 2018
Accepted: 6 April, 2018
**Corresponding Author: beongku@hongik.ac.kr
Dept. of Software & Communications Engineering, Hongik University, Korea

려하지 않고 요구사항이나 명세를 기반으로 테스트하는 방법이다. 화이트박스 테스트와는 반대로 프로그램의 상세한 동작을 검사하는 것이 아니라 입력한 값에 대한 출력이 기대한 결과인지 아닌지를 검사하는 방법이므로, 소프트웨어의 기능이 정상적으로 동작하는지 검사하는 방법이다[1-13].

소프트웨어에 숨어있는 결함을 발견하기 위해서 소프트웨어를 작동시키는 행위는 시간과 노력이 매우 많이 필요하다. 좋은 테스트 기법을 적용한다면 적은 노력으로 많은 오류를 찾아낼 수 있겠지만, 모든 경우를 테스트하여 오류가 전혀 없다는 것을 확인하기 위해서는 결국 기하급수적인 시간과 비용이 들어갈 것이다. 그렇기 때문에 단순하거나 경우의 수가 많은 테스트 케이스에 대해서는 테스트 자동화 도구가 필요하다.

본 연구에서는 윈도우 환경에서 블랙박스 테스트 기법을 사용하여 GUI 관점으로 테스트를 자동화하는 테스트 자동화 도구를 연구 및 개발하고자 한다. GUI 관점에서 입력은 키보드 입력, 마우스를 이용한 클릭, 드래그 등으로 발생할 수 있고 이에 대한 출력은 화면으로 나타날 것이다. 이 단순한 입력을 이용한 테스트 케이스는 무수히 많을 수밖에 없기 때문에 자동화 도구가 꼭 필요하다. 특히 솔루션 개발 시에 기능이 계속하여 추가되는 경우라면 모듈의 통합에 따른 치명적인 사이드 이펙트를 모두 검출하기는 불가능할 것이다. 그렇기 때문에 본 연구는 단순하고 수많은 입력으로 예기치 않은 오류가 발생하여 프로그램이 종료되는 오류나 예상치 못한 부분에서 예상했던 오류가 발생하는 경우를 찾아내는 자동화 도구를 연구 및 개발하는 것을 목표로 한다.

본 논문의 구성은, Ⅱ장에서 관련 연구를 소개한다. Ⅲ장에서는 제안된 시스템을 소개하며, 이 시스템의 기본 개념과, 주요 부분인 이미지 비교 모듈을 설명한다. Ⅳ장에서는 제안된 시스템의 성능 평가를 서술하며, 성능 평가 환경과 파라미터, 그리고 결과를 보고한다. 마지막으로 Ⅴ장에서는 제안된 시스템에 대한 결론을 맺는다.

Ⅱ. 관련 연구

1. GUI 버그 검출을 위한 블랙박스 기반의 시험[1]

이 연구는 GUI 버그를 검출하기 위해 자동화된 블랙박스 시험 방법을 제안한 것이 본 연구와 비슷했지만, 안드로이드 설치파일(apk)로부터 액티비티 정보를 파악하여 그에 대한 무작위 이벤트를 발생시키는 것이 본 연구와 방법이 달랐다. 본 연구는 설치파일 등에 대한 정적 분석에 근거한 테스트가 아닌 동적인 GUI 환경에서 무작위 이벤트를 발생시키므로 이 연구와 차이가 있다.

2. Record-Playback 기술 기반의 GUI 테스트 케이스 자동 생성[2]

이 연구는 GUI 기반의 테스트 케이스를 주제로 한 연구이다. 하지만 이 연구에서는 윈도우 이벤트를 기반으로 테스트 케이스를 생성했다는 점이 본 연구의 목적과 달랐다. 윈도우 이벤트를 한 스텝으로 놓고 이를 조합하여 테스트 케이스를 생성했던 것이다. 메뉴, 텍스트, 버튼, 윈도우 네 가지 구성으로 GUI를 분류한 점을 참고하였다.

3. 효과적인 모델 기반 안드로이드 GUI 테스팅을 위한 동일 화면 비교 기법[3]

이 연구는 GUI 상태를 정밀하게 구분하고 유효한 테스트 모델을 생성하여 안드로이드 GUI 테스트의 성능 향상 가능성을 제시한다. 본 연구에서도 GUI 상태를 저장하지만, 결과 분석용으로 사용하며, 모델 생성을 하지 않는 점이 차이가 있다.

Ⅲ. 제안된 시스템

1. 제안된 시스템의 기본 개념

제안된 시스템의 기본개념 및 동작진행 과정은 다음과 같다.

- **Step 1(목적 프로그램 지정) :** 파일 탐색기를 통하여 대상 프로그램을 선택한다.
- **Step 2(목표 범위 지정) :** 마우스로 테스트할 좌표 범위를 선택한다.
- **Step 3(옵션 지정) :** 목표 이미지, X·Y 좌표 증가 값 또는 횟수, 이벤트 대기 시간, 이미지 템플릿 매칭 임계값 등을 선택한다.
- **Step 4(테스트 매크로 실행) :** 설정 사항에 따라 마우스 매크로가 실행되며, 로그가 출력된다.
- **Step 5(결과 로그 분석) :** 저장된 로그와 캡처된 이미지를 분석한다.

2. 테스트 옵션

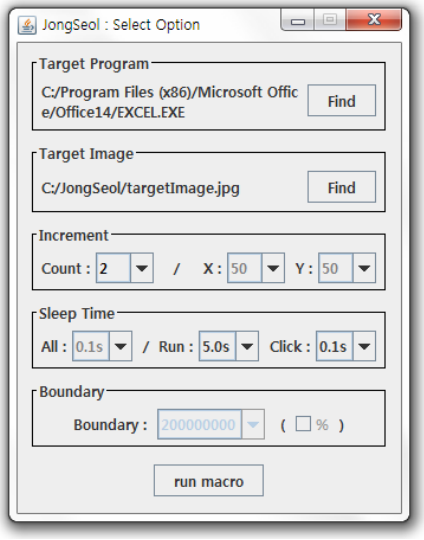


그림 1. 옵션 지정 화면
Fig. 1. Specify options screen

그림 1은 제안된 시스템의 동작진행 순서 중 Step 3 (옵션 지정)에 해당하는 부분을 구현한 화면이다. 위부터 순서대로 목적 프로그램, 목표 이미지, 좌표 증가 값 또는 횟수, 이벤트 대기 시간, 템플릿 매칭 임계값을 지정하는 입력 폼이다. 각 옵션에 대한 설명은 다음과 같다.

목적 프로그램은 파일 탐색기를 통해 선택하게 된다. 프로그램을 선택하고 나면 선택한 프로그램을 실행하여 목표 범위를 지정할 수 있도록 다이얼로그를 띄워 마우스로 좌표를 입력받은 다음, 다시 이 옵션 지정 화면을 표시한다. 좌표 증가 값 또는 횟수 옵션은 횟수로 지정하면 증가 값 옵션이 비활성화 되고, 증가 값으로 지정하면 횟수 옵션이 비활성화 된다. 이벤트 대기시간 옵션은 목적 프로그램 실행을 위한 대기 시간과 마우스 클릭을 위한 대기 시간 두 가지가 있다. All 옵션은 두 가지 대기 시간을 한 번에 지정하기 위한 편의 옵션이다. 템플릿 매칭 임계값은 2.0E8로 고정되어있다.

3. 테스팅 알고리즘

그림 2는 제안된 시스템의 테스팅 알고리즘을 설명하고 있다. 이 흐름도에서 나타나는 루프(반복 구간)는 제안된 시스템의 진행 순서 중 step 4(테스트 매크로 실행)의 테스트 스텝 1회를 의미한다.

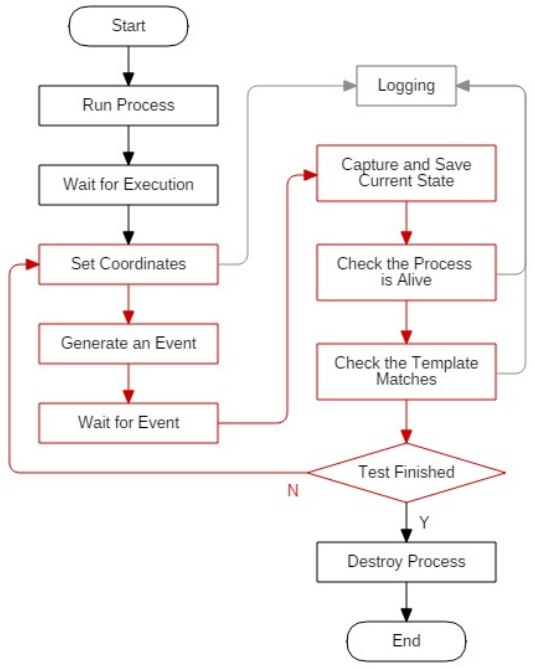


그림 2. 제안된 시스템의 흐름도
Fig. 2. Flow chart of proposed system

4. 이미지 비교 모듈

제안된 시스템의 프로그램은 Step 4(테스트 매크로 실행) 도중에 설정된 좌표 또는 횟수에 따라 마우스를 눌렀다 떼는 동작을 반복하며, 이 한 번의 테스트 스텝마다 전체 화면을 캡처하여 목표 이미지와 비교한다. 이미지 비교는 C언어 기반의 오픈소스 OpenCV를 이용하여 jpg 이미지의 'Template Matching'으로써 구현하였다.

IV. 제안된 시스템의 성능 평가

1. 성능 평가 환경

성능 평가는 Windows 7 (64-bit) 운영체제에 Java 8 Update 144와 OpenCV 3.2.0이 설치된 환경에서 진행하였다. 제안된 시스템을 구현한 java 프로그램을 jar파일로 배포하여 성능 평가를 수행하기 위한 다른 환경은 필요하지 않았다.

- Windows 7 (64-bit)
- Java 8 Update 144 (64-bit)
- OpenCV 3.2.0

2. 성능 평가 파라미터

성능평가를 위해서 사용된 성능평가 파라미터 값들은 다음과 같다.

- 목적 프로그램 : Microsoft Excel 2010
- 목적 이미지 : 셀 서식 다이얼로그 이미지
- 증가값 : 3(count), 13(x), 18(y)
- 대기 시간 : Run 5.0s, Click 0.1s
- 임계값 : 2.0E8

여기서 템플릿 매칭 임계값은 다음과 같이 설명된다.

템플릿 매칭 임계값: 성능 평가에 앞서 여러 가지 목적 이미지를 사용하여 구현된 시스템을 테스트한 결과 템플릿 매칭의 임계값은 2.0E8에서 명확하게 구분되었다. 이 경계값은 method CV_TM_SQDIFF(픽셀 차의 제곱의 합)[4] 기준으로 구현한 기준이다.

템플릿 매칭 임계값을 선정하기 위한 실험 결과를 표 1에 정리하였다. 각 이미지들에 대한 설명은 다음과 같다. phone1은 스마트폰의 어플 서랍 이미지이고, phone2도 스마트폰 어플 서랍 이미지이지만 phone1과는 다른 이미지이다. notepad-file은 윈도우 메모장의 파일 메뉴가 선택된 이미지이며 notepad-edit는 윈도우 메모장의 편집 메뉴가 선택된 이미지이다. -part가 이미지 이름 뒤에 붙어있는 이미지는 이 접미어 앞에 해당하는 이미지의 부분 이미지이다. 예를 들어 phone1-part는 스마트폰 어플 서랍 중의 아이콘 하나만의 이미지이며, notepad-edit-part는 윈도우 메모장의 편집 드롭다운 메뉴만의 이미지이다. -copy가 이미지 이름 뒤에 붙어있는 이미지는 이 접미어 앞에 해당하는 이미지의 복사본이며 훼손된 이미지이다.

이 실험의 기대 결과는 원본 이미지의 부분 이미지나 훼손된 이미지에 대하여 2.0E8보다 작은 값을 반환하는지 확인하는 것이다. 실험 결과 모든 변수에 대하여 기대에 충족하는 결과를 얻었고, 정상적으로 목표 이미지를 검출했다는 의미로 결과를 true와 false로 표 1에 표기하였다.

3. 성능 평가 결과

테스트 수행이 끝나면 테스트 결과는 그림 3과 같은 로그 파일로 저장된다. 각 줄마다 시간, 순서, 좌표, 프로세스 상태, 목적 이미지 검출 여부를 탭에 맞추어 표시하고, 목적 이미지가 검출된 스텝은 가장 끝에 “detected!”라는 문구로 명확히 확인할 수 있게 표시하도록 했다.

표 1. 템플릿 매칭 임계값 선정을 위한 실험 결과
Table 1. Experimental results for template matching threshold selection

원본 이미지	템플릿 이미지	반환값	결과
phone1	phone1	1.071000e+04	true
phone1	phone1-part	1.203998e+06	true
phone1	phone1-part-copy	1.060109e+08	true
phone1	phone2	1.499356e+10	false
phone1	notepad-file-part	2.160244e+09	false
phone1	notepad-edit-part	4.897771e+09	false
phone1-part	phone1-part	0.000000e+00	true
phone1-part	phone1-part-copy	1.045222e+08	true
phone1-part-copy	phone1-part	1.045222e+08	true
phone1-part-copy	phone1-part-copy	0.000000e+00	true
phone2	phone1	1.499356e+10	false
phone2	phone1-part	2.394003e+08	false
phone2	phone1-part-copy	2.875435e+08	false
phone2	phone2	0.000000e+00	true
phone2	notepad-file-part	2.030341e+09	false
phone2	notepad-edit-part	4.777647e+09	false
notepad-file	phone1-part	3.473923e+09	false
notepad-file	phone1-part-copy	3.811994e+09	false
notepad-file	notepad-file	3.180200e+04	true
notepad-file	notepad-file-part	6.057820e+05	true
notepad-file	notepad-edit-part	3.702349e+08	false
notepad-file-part	notepad-file-part	0.000000e+00	true
notepad-edit-part	phone1-part	3.558118e+09	false
notepad-edit-part	phone1-part-copy	3.904361e+09	false
notepad-edit-part	noteapd-file-part	1.779960e+08	false
notepad-edit-part	notepad-edit-part	0.000000e+00	true

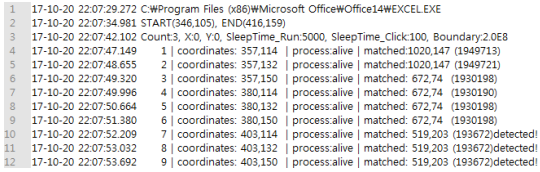


그림 3. 로그 파일
Fig. 3. Log File

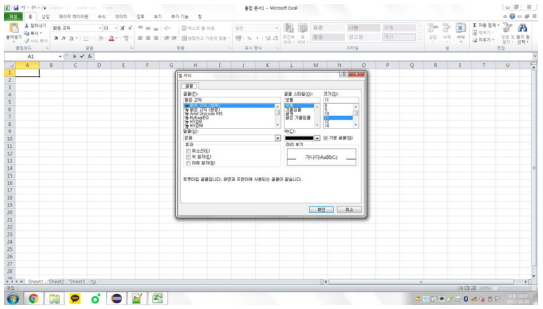


그림 4. 캡처된 이미지에서 지정한 이미지를 찾아냄
Fig. 4. Find the specified image in the captured image

각 스텝마다 상태를 캡처한 이미지가 저장된다. 목적 이미지가 검출되었다면 위의 그림 4와 같이 박스로 표시하여 나타낸다.

V. 결론

본 연구에서 있었던 제한사항은 두 가지였다. 첫 번째로, 제안된 시스템은 로그와 캡처된 이미지를 통해 결과를 분석할 수는 있다. 하지만 기존에 있던 입력 데이터와 관련한 블랙박스 테스트 케이스 설계 기법(경계값 또는 상태 등에 기반을 둔 설계 기법)에 따라 정통적인 테스트 케이스 목록을 작성하지는 않는다. 따라서 테스트 커버리지를 판단하기가 모호했다. 두 번째로, 오류 발생 상황을 예측할 수 없다고 가정했을 때, 오류 검출을 위한 목표 이미지를 정의하기가 어려웠다.

테스트 케이스를 별도로 작성하지 않는 것은 퍼징과 의미가 유사하다. 퍼징이란 프로그램에 유효한 예상치 않은, 또는 무작위 데이터를 입력하여 예상하지 못한 오류를 발생시키는 것이다[5]. 테스트 케이스를 생성하지 않는 대신에 제안된 시스템을 이용하여 입력 데이터를 무작위로 발생시키면 예상하지 못한 오류를 확인할 수도 있다. 만약 목표 이미지를 프로그램 응답 없음 오류 등의 이미지로 지정한다면 이 퍼징이라는 의미가 더욱 뚜렷해질 것이다.

제안된 시스템을 퍼징이 아닌 블랙박스 테스트 도구로 이용한다면 디버그 도구로 활용할 수도 있다. 어떤 종류의 오류 상황에 특정 메세지 다이얼로그가 출력되도록 디버그를 위한 경고를 지정한다면 제안된 시스템은 이 경고 상태를 찾아낼 수 있으므로 예상할 수 있지만 스텝을 알 수 없는 오류를 검출할 수 있다.

향후 연구 계획으로는, GUI 이벤트 발생 순서에 대해 조합이 무수히 많으므로 효율적인 알고리즘 추가 개발을 통해 테스트 수행 능력을 향상 시킬 것이다. 테스트 자동화에 중점을 둔다면 현재 화면 및 프로세스 상태에 따라 테스트 방향을 자동으로 바꾸는 기능이 필요해질 것이다. 또한, 다양한 목적 프로그램을 대상으로 실용성에 대해서 평가하고자 한다.

References

[1] Jemin Lee, Hyungshin Kim, “A Black-Box based Testing for GUI Bug Detection,” Journal of KIISE, 41(12), pp.1013-1017, December 2014.

[2] JungGyuew Lee, HyeonSoo Kim, SeunGhak Kuk, DaeWan Cho, “Record-Playback based Automatic test case generation for GUI test,” Proc. of KCC 2007, vol.34, no.1(B), pp.96-100, June 2007.

[3] Youngmin Baek, Gwangui Hong, Cheonghyun Lee, Doo-Hwan Bae, “A GUI State Comparison Technique for Effective Model-based Android GUI Testing,” Journal of KIISE, JOK, vol.42, no.11, pp. 1386-1396, 2015.

[4] OpenCV,“Template Matching,” https://docs.opencv.org/3.0-last-rst/doc/tutorials/imgproc/histograms/template_matching/template_matching.html, December 2014.

[5] Wikipedia, “Fuzzing,” https://ko.wikipedia.org/wiki/%ED%8D%BC%EC%A7%95, March 2016.

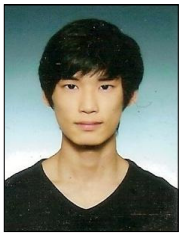
[6] Jung Gyw Lee, Seung Hak Kuk, Hyeon Soo Kim, “Test Cases Generation Method for GUI Testing with Automatic Scenario Generation,” Journal of KISS : Software and Applications 36(1), pp.45-53, January 2009.

[7] Atif M. Memon, Martha E. Pollack and Mary Lou Soffa, Hierarchical GUI Test Case Generation Using Automated Planning, IEEE Transaction on Software Engineering, vol.27, no.2, pp.144-155, February 2001.

[8] Atif M. Memon, GUI Testing: Pitfalls and Process, IEEE Computer, pp.90-91, August 2002.
[9] Jessica Chen and Suganthan Subramaniam, Specification-based Testing for GUI-based Application, Software Quality Journal, vol.10, pp.205-224, 2002.
[10] Li, Kanglin, Mengqi Wu, Effective GUI testing automation: Developing an automated GUI testing tool, John Wiley & Sons, 2006.
[11] Myers, Glenford J., Corey Sandler, and Tom Badgett, The art of software testing, John Wiley & Sons, 2011.
[12] Soeui Kim, Duri Choi, Beongku An, "Detection and Prevention Method by Analyzing Malignant Code of Malignant Bot," JIIBC, Vol. 13, No.2, pp.199-207, April 2013.
[13] Sungjik Choi, Minji Kim, Jeungwook Han, Beongku An, "Android Based Mobile Student Identity Card," JIIBC, Vol. 13, No.2, pp. 209-215, April 2013.

저자 소개

정 범 진(정회원)



• 2018년 : 홍익대학교 컴퓨터정보통신공학과 (BS)
<주관심분야 : SW 아키텍처, 테스트 자동화>

이 정 우(정회원)

• 2018년 : 홍익대학교 컴퓨터정보통신공학과 (BS)
<주관심분야 : SW 품질관리, 테스트 자동화>

홍 창 완(정회원)

• 2018년 : 홍익대학교 컴퓨터정보통신공학과 (BS)
<주관심분야 : SW 품질관리, 테스트 자동화>

안 병 구(종신회원)

• 1988년 : 경북대학교 전자공학과 (B.S)
• 1996년 : (미)New York University (Polytechnic) Dept. of Computer and Electrical Eng, (M.S)
• 2002년 : (미)New Jersey Institute of Technology(NJIT), Dept. of Computer and Electrical Eng. (Ph.D)
• 1989년 ~ 1994년 : 포항산업과학기술연구원(RIST), 선임연구원
• 2012년 : 대한전자공학회 컴퓨터소사이어티 회장
• 2003년 ~ 현재 : 홍익대학교 소프트웨어융합학과 교수
<주관심분야 : Mobile Wireless Networks & Communications, Ad-hoc & Sensor Networks, Multicast Routing, QoS Routing, VLC, Cognitive Radio Networks, Energy Harvesting, Physical Layer Security, Mobile Cloud Computing, Cross-Layer Technology, 5G, NOMA, SWIPT, Network Coding, Cooperative Communication, Bioinformatics>

※ This work was supported by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (Grant No. 2016R1D1A1B03934898) and by the Leading Human Resource Training Program of Regional Neo industry Through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future planning (Grant No. 2016H1D5A1910577).