한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

제 21 권 제 1 호
Vol. 21 No. 1

SOFTWARE ENGINEERING SOCIETY

소프트웨어공학회원고 KIPS

2019

제 21 회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 21th Korea Conference on Software Engineering (KCSE 2019)

- **일시: 2019년 1월 28일(월) ~ 1월 30일(수)**
- **장소: 강원도 평창 한화리조트(휘닉스파크점)**

주최: 한국정보과학회, 한국정보처리학회

주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학연구회

후원: ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, SOLUTIONLINK, WISESTONE
글로벌소프트웨어캠퍼스㈜, ㈜비트컴퓨터, T3Q㈜,
㈜다한테크, 소프트웨어공학엑스퍼트그룹㈜,
슈어소프트테크㈜, ㈜에스피아이디, STA 테스팅컨설팅㈜,
TTA 소프트웨어시험인증연구소, 신뢰적지능형 CPS 연구단,
풀스택다중언어 SW 검증 및 디버깅연구단

초대의 글

소프트웨어공학인의 축제인 제 21 회 한국 소프트웨어공학 학술대회(KCSE 2019) 참가자 여러분을 환영합니다.

기업, 연구소 및 학계에서 활동하고 계신 소프트웨어공학 분야 전문가들의 모임인 한국정보과학회 소프트웨어공학 소사이어티와 한국정보처리학회 소프트웨어공학연구회는 소프트웨어공학 기술의 발전 및 적용확산을 위하여 산, 학, 연과의 협력으로 한국 소프트웨어공학 학술대회를 개최하게 되었습니다.

제 4 차 산업혁명의 시대에 인공 지능, 사물 인터넷, 빅데이터, 모바일 등의 신기술이 결합되어 실세계를 네트워크로 연결하고 사물을 지능화하기 위해서, 소프트웨어가 그 핵심이 됨에 따라 이번 학술대회에서는 특히 “초연결·초지능·초융합을 위한 소프트웨어공학 기술”을 주제로 하여 소프트웨어공학 전분야에 걸쳐 각계에서 제출한 66 편의 논문이 발표됩니다. 특히, 우수국제학술대회 논문 3 편과 우수국제저널 논문 6 편을 초청하여 프로그램을 구성했습니다. 또한 소프트웨어 테스팅, 확률적 프로그래밍, 블록체인, SW 위험원/결함 분석, 데이터 사이언스 등의 최근 기술 동향과 소프트웨어공학의 핵심기술을 학습할 수 있는 여섯 개의 튜토리얼이 준비되어 있습니다. 그리고 소프트웨어 저장소 마이닝을 이용한 결함 검출, 소프트웨어 공수/결함 예측 등의 최근 연구 주제에 대하여 세 분의 신임교수께서 세미나를 해 주십니다. 아울러 연구개발 품질관리, 소프트웨어 안전성, 사물 경제 시대 등 소프트웨어 공학의 중요한 이슈와 미래방향에 대하여 세 분의 저명인사께서 기조강연을 해 주십니다.

본 학술대회가 소프트웨어공학의 학문적 발전과 소프트웨어 산업기술 발전의 장이 되고, 학술교류 및 기술 협력을 위한 활발한 토론장이 될 수 있도록 여러분의 적극적인 참여를 부탁드립니다.

본 KCSE 2019 행사를 위해 수고해 주신 조직위원회와 학술위원회 위원들과 여러 후원 기관에 깊이 감사드립니다.

한국정보과학회 소프트웨어공학 소사이어티 회장 이병정
한국정보처리학회 소프트웨어공학연구회 운영위원장 김정아

학술대회 준비 위원회

공동대회장: 이병정 교수(서울시립대), 김정아 교수(카톨릭관동대)

조직위원장: 홍장의 교수(충북대)
조직위원: 백종문 교수(KAIST), 유준범 교수(건국대), 이선아 교수(경상대),
이정원 교수(아주대), 이지현 교수(전북대), 이찬근 교수(중앙대),
한종대 교수(상명대), 김정훈 이사(와이즈스톤), 민상윤 대표(솔루션링크),
정효택 박사(ETRI), 김성우 대표(글로벌소프트웨어캠퍼스),
송인옥 부장(비트컴퓨터), 박병훈 대표(티쓰리큐)

학술위원장 김순태 교수(전북대)
학술위원: 고덕윤 교수(서강대), 고인영 교수(KAIST), 김문주 교수(KAIST),
김민성 박사(현대자동차), 김진태 대표 (SEEG), 남재창 교수(한동대),
류덕산 교수(전북대), 박수진 교수(서강대), 박용범 교수(단국대),
배경민 교수(포항공대), 서영석 교수(영남대), 서주영 교수(아주대),
염근혁 교수(부산대), 유신 교수(KAIST), 유철중 교수(전북대), 윤회진 교수(협성대),
이관우 교수(한성대), 정우성 교수(서울교대), 지은경 교수(KAIST),
채흥석 교수(부산대), 최윤자 교수(경북대), 홍신 교수(한동대)

문의사항 연락처

학술대회 홈페이지 : http://www.sigsoft.or.kr/KCSE2019/
조 직 : 홍장의 교수 (jehong@cbnu.ac.kr 043-261-2261)
학 술 : 김순태 교수 (stkim@jbnu.ac.kr 063-270-4788)

KCSE 2019 프로그램

1 월 28 일 (월)				
시간	**행 사 내 용**			
12:00–13:00	**KCSE 2019 등록**			
	튜토리얼 T1 좌장: 이정원(아주대) 장소: 세미나실 1	**튜토리얼 T2** 좌장: 유신(KAIST) 장소: 세미나실 2	**튜토리얼 T3** 좌장: 고덕윤(서강대) 장소:세미나실 3	**워크숍: 신뢰적 지능형 CPS 연구단** 장소: 세미나실 4 (09:00-16:10)
13:00-14:30 (90 분)	소프트웨어 테스팅 국제표준 (ISO/IEC/IEEE 29119)의 핵심 내용 이해와 활용 방안 권원일 대표(STA 테스팅컨설팅)	범용 통계 추론을 위한 확률적 프로그래밍 (Probabilistic Programming) 양홍석 교수(KAIST)	Hyperledger 를 이용한 응용 프로그램 개발 박용범 교수(단국대)	
14:30-14:40	**휴식**			
	튜토리얼 T4 좌장: 김순태(전북대) 장소: 세미나실 1	**튜토리얼 T5** 좌장: 최윤자(경북대) 장소:세미나실 2	**튜토리얼 T6** 좌장: 배경민(포항공대) 장소:세미나실 3	
14:40-16:10 (90 분)	소프트웨어 안정성을 위한 위험원 분석기법 권기현 교수(경기대)	프로그램 분석을 사용한 결함 검출 (Bug Detection via Program Analysis) 류석영 교수(KAIST)	데이터 사이언스와 소프트웨어 공학 - 반도체 분야 중심으로 박찬진 상무(SK Hynix)	
16:10-16:20	**휴식**			
	개회식 장소: 그랜드홀 2		사회: 홍장의 조직위원장(충북대)	
16:20-16:40 (20 분)	개회사: 이병정 회장(한국정보과학회 소프트웨어공학 소사이어티) 김정아 운영위원장(정보처리학회 소프트웨어공학연구회)			
	기조강연 I 장소: 그랜드홀 2		사회: 홍장의 조직위원장(충북대)	
16:40-17:30	기조강연 1:연구원 사례 중심의 R&D(연구개발) 품질관리 - 정효택 박사(ETRI)			
	신임교수 초청 세미나 1 좌장: 한종대(상명대) 장소: 세미나실 1	**신임교수 초청 세미나 2** 좌장: 지은경(KAIST) 장소: 세미나실 2	**신임교수 초청 세미나 3** 좌장: 이선아(경상대) 장소: 세미나실 3	
17:30-18:20	마이닝 소프트웨어 레포지토리를 이용한 버그검출도구 개선 남재창 교수(한동대)	데이터 분할법에 의한 소프트웨어 공수 예측 정확도 향상 서영석 교수 (영남대)	클래스 불균형을 고려한 교차프로젝트 결함예측 류덕산 교수 (전북대)	
18:20-19:10	**석 식**			

1 월 29 일 (화)

시 간	행 사 내 용				
	논문 발표 A				
	A1: 보안 및 안전 1 좌장: 유철중(전북대) 장소: 그랜드홀 2	**A2: SW 테스팅 1** 좌장: 고인영(KAIST) 장소: 세미나실 1	**A3: 블록체인** 좌장: 고덕윤(서강대) 장소: 세미나실 2	**A4: 유지보수** 좌장: 서영석(영남대) 장소: 세미나실 3	
09:20-11:00 (100 분)	An empirical study of collaborative model and its security risk in Android [초청논문: JSS'18] Ajay Kumar Jha, 이우진(경북대)	크로스 플랫폼 애플리케이션의 테스트 자동화 방안 [일반논문] 황준선, 최은만(동국대)	스마트 컨트랙트 및 DApp 신뢰성 향상을 위한 단계별 체계적인 테스트 기법 [우수 일반논문] 민경식(서울시립대), 이정원(아주대), 이병정(서울시립대)	MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C [초청논문: FSE'18] 홍성준, 오학주(고려대)	**워크숍: 풀스택 다중언어 SW 검증 및 디버깅 연구단** 장소: 세미나실 4 (09:00-15:40)
	시나리오 테이블을 이용한 STPA 에서의 사고 원인 식별 [우수 일반논문] 양현수, 권기현(경기대)	Automatic Data Flow Analysis to Generate Test Cases from Activity Diagrams [단편논문] Aman Jaffari, 이지현, 유철중(전북대)	GAS 절감이 가능한 스마트 컨트랙트 코드 패턴 분석 방법 [단편논문] 정다은, 박수용(서강대)	어휘적 특징에 기반한 프로그램 자동 패치 검색 공간 축소 [우수 단편논문] 안가빈, 유신(KAIST)	
	차량 수준 Hazard Analysis Risk Assessment 활동의 취약점과 실용적 해결방안 연구 [단편논문] 정효윤, 민상윤(KAIST)	빅데이터 분석용 DBMS 를 위한 테스트 자동화 프레임워크 설계 [산업체 논문] 임우섭, 송옥수, 김현정(한국정보통신 기술협회)	솔리디티 스마트 컨트랙트의 스크립트 기반 시나리오 테스트 [우수 단편논문] 문현아, 박수용(서강대) 최광훈(전남대)	MDA 기반 태양광 에너지 모니터링 시스템의 유지보수를 위한 비용 감소 방안 사례 [단편논문] 장우성, 김영철(홍익대)	
	순환신경망을 활용한 맥락 데이터 기반 사용 자 인증 기법 [단편논문] 남상진, 김순태, 신정훈(전북대)	Precise concolic unit testing of C programs using extended units and symbolic alarm filtering [초청논문: ICSE'18] 김윤호, 김문주(KAIST)	시장도매인 위탁 판매를 위한 블록체인 기반 시스템 설계 [학부생 논문] 홍준기, 김순태(전북대)	TF-IDF 알고리즘과 API 정보를 활용한 윈도우 실행파일 분류 [학부생 논문] 문재영, 강민구, 김순태, 김강연, 최상철(전북대)	
11:00-11:10	**휴식**				

1 월 29 일 (화)

시 간	행 사 내 용				
	논문 발표 B				
	B1: 레포지토리 마이닝 좌장: 한종대(상명대) 장소: 그랜드홀 2	**B2: SW 품질 1** 좌장: 남재창(한동대) 장소: 세미나실 1	**B3: SW 검증 1** 좌장: 이선아(경상대) 장소:세미나실 2	**B4: 스마트 시스템** 좌장: 서영석(영남대) 장소: 세미나실 3	
11:10-12:30 (80 분)	RNN 을 이용한 이슈 분류 [일반논문] 조희태, 이선아(경상대) 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 향상 [일반논문] 최현우, 유신(KAIST) 순환 인공 신경망을 이용한 코드 변경 추천 [단편논문] 이재준, 강성원(KAIST) 이선아(경상대) 중복된 이슈 보고서 요약 [학부생 논문] 이현권, 김찬규, 이선아(경상대)	BIMM 에 기반한 소프트웨어 산업의 비즈니스 혁신성 가이드를 위한 적용사례 [단편논문] 박영식, 김영철(홍익대) 항공기 프로펠러 고장진단을 위한 압력센서 비교 [단편논문] 송슬기, 김은진, 이선아(경상대) Automatic Usage Profiling 을 통한 초기 앱 실행 속도 개선 방법 [우수 산업체 논문] 채향석(LG 전자) 백종문(KAIST) 애자일 전환을 위한 핵심요소 [산업체 논문] Hohaeng Cho(STA 컨설팅)	Dependability-enhanced unified modeling and simulation methodology for Critical In frastructures [초청논문: IST'18] 김희수(리얼타임비주얼), 이석원(아주대) 모델 기반 멀티타스크 프로그램의 API 호출 안전성 검증을 위한 오경보 식별 및 성능 향상 기법 [우수 일반논문] 김동우, 최윤자(경북대) 소프트웨어 제품 라인 개발을 위한 가변성 모델의 무모순성 검증 [단편논문] 한영훈, 강성원, 이동민(KAIST, 박병훈(T3Q)	패혈증 진단용 REBA 검사 자동화를 위한 영상인식 모바일 어플리케이션 개발 [일반논문] 류원철, 황상원, 서강원, 남영광(연세대) 스마트 빌딩을 위한 b.IoT 솔루션과 휴먼센싱 기술 [단편논문] Hunjung Lim(삼성전자) 대사증후군 예방을 위한 EMC 프로그램 개발 [단편논문] 이수경(계명대) LoRa 기반의 치매환자를 위한 안전관리 시스템 [학부생 논문] 정재호, 김홍준, 최민철, 이보경(한국산업기술대)	**워크숍: 풀스택 다중언어 SW 검증 및 디버깅 연구단** 장소: 세미나실 4 (09:00-15:40)
12:30-13:40	**중식**				

<table>
<tr><td></td><td colspan="4">논문 발표 C</td><td></td></tr>
<tr><td></td><td>C1: SW 품질 2
좌장: 백종문(KAIST)
장소: 그랜드홀 2</td><td>C2: 분석 및 평가
좌장: 이찬근(중앙대)
장소: 세미나실 1</td><td>C3: SW 검증 2
좌장: 남재창(한동대)
장소: 세미나실 2</td><td>C4: 설계 및 모델링
좌장: 황상원(연세대)
장소: 세미나실 3</td><td></td></tr>
<tr><td>13:40-15:40
(120 분)</td><td>Automatically generating search heuristics for concolic testing [초청논문: ICSE'18]
차수영, 오학주(고려대)

무기체계 소프트웨어 신뢰성 통합관리시스템 소개 [산업체 논문]
김태현, 임종일, 노성규, 백옥현, 이태호(국방과학연구소)

블록체인 시스템의 품질모델 제안 [후원 산업체 논문]
고재정 (와이즈스톤)

서비스 기반 V2X 소프트웨어 환경에서의 서비스 소비 품질 최적화 방법 [최우수 일반논문]
문형철, 백경덕, 고인영(KAIST)</td><td>Engine Control Unit (ECU) 실시간 객체 지향 모델의 능동 객체 기반 우선순위 전파 멀티 코어 멀티 태스킹 [일반논문]
김경률, 김세화(한국외국어대)

캡슐내시경 영상 데이터 중심 소프트웨어 개발을 위한 데이터 셋 품질 평가 매트릭 [일반논문]
박예슬(아주대), 이병정(서울시립대), 이정원(아주대),

Generative Adversarial Network 을 이용한 소프트웨어 결함 예측 [단편논문]
이주혜, 진영인, 남재창(한동대)

REASSURE: Requirements elicitation for adaptive socio-technical systems using repertory grid [초청논문: IST'17]
Sangeeta Dey, 이석원(아주대)</td><td>C 언어로 작성된 OSEK / VDX OS 모델을 사용한 임베디드 제어 소프트웨어 검증 [일반논문]
양송, 김동우, 최윤자(경북대)

고품질 무기체계 소프트웨어를 위한 신뢰성 시험 환경 구축에 관한 연구 [우수 산업체 논문]
최민관, 박다운, 국승학(국방과학연구소)

사이버-피지컬 시스템의 상호운용성 상충관계 분석에 관한 연구 [단편논문]
김의섭, 유준범(건국대)

A configurable V&V framework using formal behavioral patterns for OSEK/VDX operating systems [초청논문: JSS'18]
최윤자(경북대)</td><td>Spiking Neural Network 의 다양한 신경세포 모델을 지원하는 프로그래밍 방법론 [단편논문]
이헌준, 김장우(서울대)

고신뢰성 내장형 무기체계 소프트웨어를 위한 실행 가능한 모델링 언어 [산업체 논문]
배정호, 구봉주(국방과학연구소)

빅데이터 인공지능 통합 플랫폼 아키텍처 [후원 산업체 논문]
박병훈(T3Q)

Collaborative Design Conflicts: Costs and Solutions [초청논문: IEEE Software'18]
방재영(카카오)</td><td>워크숍: 풀스택 다중언어 SW 검증 및 디버깅 연구단

장소: 세미나실 4 (09:00-15:40)</td></tr>
<tr><td>15:40-16:00</td><td colspan="4">휴식</td><td></td></tr>
<tr><td></td><td colspan="4">기조강연 II
장소: 그랜드홀 2
사회: 김순태 학술위원장 (전북대)</td><td></td></tr>
<tr><td>16:00-16:50</td><td colspan="4">기조강연 2: 4 차 산업혁명과 소프트웨어 안전, 그리고 우리의 사명 - 민상윤 대표 (솔루션링크)</td><td></td></tr>
<tr><td></td><td colspan="4">기조강연 III
장소: 그랜드홀 2
사회: 이병정 대회장 (서울시립대)</td><td></td></tr>
<tr><td>16:50-17:40</td><td colspan="4">기조강연 3: 사물 인터넷을 넘어 사물 경제 시대로의 변화 - 박수용 교수 (서강대)</td><td></td></tr>
<tr><td>17:40~18:00</td><td colspan="4">휴식</td><td></td></tr>
</table>

18:00- 21:00	**우수논문상, 공로상, 감사장 수여식** **만찬** 장소: 그랜드홀 1	사회: 홍장의 조직위원장 (충북대) 사회: 한종대 교수 (상명대)

1 월 30 일 (수)

시 간	행 사 내 용			
	논문 발표 D			
	D1: SW 품질 2 좌장: 홍신(한동대) 장소:그랜드홀 2	**D2: SW 테스팅 2** 좌장: 유준범(건국대) 장소:세미나실 1	**D3: SW 아키텍처** 좌장: 황상원(연세대) 장소: 세미나실 2	**D4: 도구 및 개발환경** 좌장:이정원(아주대) 장소:세미나실 3
9:20-10:40 (80 분)	Location-based Web Service QoS Prediction via Preference Propagation to address Cold Start Problem [초청논문: TSC'18] 류덕산(전북대) 자율 군집 주행에서 그룹이탈 경보시스템의 Wright 기반 명세 [학부생 논문] 윤이진, 남승우, 홍장의(충북대) 차량용 ECU 소프트웨어의 모델 기반 개발 시 백투백 비교 테스트 방법 [산업체 논문] 김자명, 채현아, 최병주(이화여자대), 장훈(현대자동차) 오픈소스 소프트웨어 저장소에서 결함 수정 코드 수집 방안 [학부생 논문] 임성빈, 양수진, 남재창(한동대)	병행성 결함 탐지 도구에 대한 실험적 연구 [단편논문] 박창선, 박지현, 채현아, 최병주 (이화여자대) OpenDS 를 이용한 자율주행소프트웨어 시뮬레이션 [일반논문] 윤유상, 이채은, Nazakat Ali, 홍장의(충북대) 교차 프로젝트 기반 토픽 모델링과 네이브 베이지안 다항분포을 이용한 버그 리포트의 우선순위 예측 [단편논문] 양근석, 김재은(서울시립대), 이정원(아주대), 이병정(서울시립대) Testing_AI_Systems [산업체 논문] Stuart Reid, 권원일, 최영재 (STA 테스팅컨설팅)	기계학습 기반 자율 제어 시스템의 참조 아키텍처 [최우수 일반논문] 송명호, 김수동(숭실대) 휘처모델과 태스크 유사도 기반 추천 시스템을 이용한 R&D 문서 재사용 지원 [일반논문] 남승우, 혼다네스, 홍장의(충북대) 모놀리식 애플리케이션의 컨테이너 기반 마이크로서비스 전환을 위한 스크립트 생성 방법 [최우수 단편논문] 김대호, 운동규, 박준석, 염근혁(부산대) 시스템 오브 시스템즈 협업 정책 구성을 위한 상향식 기법 제안 [단편논문] B. Zelalem Mihret, 지은경, 백영민, 배두환(KAIST)	3D 대화 시나리오 저작도구 개발 [단편논문] 고연정, 신동춘, 지민성, 정윤식(콜마인) 대인관계 개선 VR 콘텐츠 생성을 위한 저작도구 설계와 구현[단편논문] 심근웅, 윤인호, 임시연, 이양민, 이재기(동아대) 시뮬레이션-이력 데이터 분석 서비스 프레임워크 설계 및 구현 [학부생 논문] 정현태, 김성현, 서영균(경북대) 동의어 치환을 이용한 심층 신경망 모델의 테스트 데이터 생성 [우수 학부생 논문] 이민수, 이찬근(중앙대)
10:40-10:50	**휴식**			

1 월 30 일 (수)			
시 간	**행 사 내 용**		
	논문 발표 E		
	E1: 자동화 좌장: 지은경(KAIST) 장소: 세미나실 1	**E2: 보안 및 안전 2** 좌장: 유준범(건국대) 장소: 세미나실 2	**E3: 패턴 및 빅데이터** 좌장:배경민(포항공대) 장소:세미나실 3
10:50-12:10 (80 분)	위치 분석 기반 통합 클라이언트-서버 프로그램 컴파일 방법 구현 [단편논문] 김가영, 최광훈(전남대) 창병모(숙명여자대) 테스트케이스의 제어흐름기반 프로그래밍 실패 피드백 시스템 설계 [우수 단편논문] 이성희, 김덕엽, 서강복, 이우진(경북대) System Test 자동화 도구 제품화를 위한 품질 개선 [산업체 논문] 하승우(LG 전자), 백종문(KAIST) 강화학습을 적용한 주행모드 변화에 따른 하이브리드 자동차 배터리 SOC 밸런싱 연구 [학부생 논문] 김영균, Sangeeta Dey, 이석원(아주대)	보안성을 고려한 경량 크래시 리포트 생성 기법 [단편논문] 장동민, 이수준, 장유원, 정호현, 이은석(성균관대) 위해도 분석 결과의 효과적인 확인을 위한 추적성 기반 위해도 모델 [단편논문] 정세진, 유준범(건국대) 안전 필수 시스템의 ISO/IEC 16085 기반 위험 분석 방법 비교 [단편논문] 이현호, 최현재, 채흥석(부산대) 행정안전부 소프트웨어 보안 취약점 진단기준과 Java 웹 어플리케이션 대상 오픈소스 보안 결함 검출기 검출대상의 총체적 비교 [우수 학부생 논문] 이재훈, 최한솔, 홍신(한동대)	선형 SVM 과 Random Forest 를 이용한 DNA 메틸화 영역 예측 [단편논문] 이충근, 한경숙(인하대) 버그 검출 규칙 개선을 위한 코드 문맥 수집 방법에 대한 연구 [단편논문] 최윤호, 김석진, 김유진, 권현우, 남재창(한동대) 수면 데이터 분석 플랫폼 설계 및 구현 [학부생 논문] 전재민, 김성현, 서영균(경북대) 의료 영상 데이터 중심 소프트웨어 개발을 위한 학습 노이즈 영상 분류 기법 [학부생 논문] 유동연, 박예슬(아주대), 이병정(서울시립대),이정원(아주대),
12:10-12:40 (30 분)	**폐회식** 장소: 그랜드홀 2		사회: 홍장의 조직위원장(충북대)

KCSE 2019 튜토리얼

튜토리얼 T1: 소프트웨어 테스팅

- 일시: 1월 28일(월) 13:00~14:30
- 장소: 세미나실 1
- 제목: 소프트웨어 테스팅 국제표준(ISO/IEC/IEEE 29119)의 핵심 내용 이해와 활용 방안
  (Core of software testing international standard, ISO/IEC/IEEE 29119, and its application in the SE classes)
- 연사: 권원일 대표(STA테스팅컨설팅)
- 튜토리얼 초록:

  소프트웨어(SW) 테스팅 국제표준은 IEEE 829, BS 7925 등의 표준을 통폐합한 국제표준이다. 이로 인해 SW 테스팅은 새로운 전문성의 시대에 접어들었고, 그에 발맞춰 해당 국제표준의 이해가 SW 공학도에게 필수적이다. 본 튜토리얼을 통해 소프트웨어 테스트 국제 표준인 ISO/IEC/IEEE 29119 에 대해 전반적인 개념을 습득하고 실무에 적용할 수 있는 방법을 알아본다. 또한 SW 공학 관련 수업(강의) 중 SW 테스팅 내용을 다룰 때 국제표준을 활용하는 손쉬운 방법을 모색해 본다. 마지막으로 ISO/IEC/IEEE 29119 Certified Tester 자격증에 대해 간략히 알아보고, 시험 응시에 필요한 핵심적인 내용의 학습방법과 해당 자격증의 수업 활용에 대해 함께 생각해 보는 시간을 갖는다.
- 약력:
  - 2004 년~현재 STA 컨설팅 대표이사
  - 2000~2002 년 한국전자통신연구원(ETRI) SW 시험인증센터 연구원
  - 2000 년 KAIST, IT 경영학 석사
- 연구분야: 소프트웨어 테스팅(AI 테스팅, 테스팅 프로세스 등), 소프트웨어 안전, 애자일 개발 방법론

튜토리얼 T2: 확률적 프로그래밍

- 일시: 1월 28일(월) 13:00~14:30
- 장소: 세미나실 2
- 제목: 범용 통계 추론을 위한 확률적 프로그래밍 (Probabilistic Programming)
- 연사: 양홍석 교수 (KAIST 전산학부)
- 튜토리얼 초록:

  기계 학습 모델을 쉽게 표현할 수 있는 프로그래밍 언어를 만들고, 이러한 언어로 작성한 모델에 사용할 수 있는 범용 통계 추론 알고리즘을 만들자는 것이 확률적 프로그래밍의 기본 아이디어입니다. 확률적 프로그래밍이 제대로 발전한다면, 추론 알고리즘을 딱히 개발할 필요가 없어져서, 초보자도 손쉽게 기계 학습 모델을 만들어 데이터를 분석할 수 있고, 전문가는 더욱 복잡한 모델을 개발하고 사용할 수 있습니다. 이러한 장점 때문에, 기계 학습, 통계학, 확률론, 프로그래밍 언어, 컴파일러 등 다양한 분야의 연구자들이 현재 관심을 가지고 확률적 프로그래밍을 연구하고 있습니다. 특히 최근에는 심층 신경망과 확률적 프로그래밍의 장점을 결합하는 연구가 활발하게 진행되고 있고, 이러한 연구 결과에 기반한 Edward 나 Pyro 와 같은 확률적 프로그래밍 언어가 개발되고 있는 상황입니다. 본 강연을 통해 이러한 확률적 프로그래밍과 현재 연구 동향을 소개하려고 합니다. 또한 제가 수행하는 연구에 대해서도 간략하게 설명할 계획입니다. 특히 확률적 프로그래밍 언어 고유의 오류를 찾아내는 프로그램 코드 분석 기술에 대해 설명할 예정입니다.
- 약력:
  - 2017 년 ~ 현재, KAIST 전산학부 정교수
  - 2011 년 ~ 2017 년, 옥스포드 대학 전산학과 부교수/정교수
  - 2006 년 ~ 2011 년, 퀸메리 런던대학 전산학과 조교수
  - 2003 년 ~ 2006 년, 서울대학교 박사후연구원

- 2001 년 ~ 2003 년, KAIST 박사후연구원
- 2001 년, UIUC 전산학과 박사

연구분야: 확률적 프로그래밍, 프로그래밍 언어, 프로그램 분석, 기계 학습, 확률론

튜토리얼 T3: Hyperledger 블록체인

- 일시: 1월 28일(월) 13:00~14:30
- 장소: 세미나실 3
- 제목: Hyperledger를 이용한 응용 프로그램 개발
- 연사: 박용범 교수 (단국대학교)
- 튜토리얼 초록:

  4 차 산업혁명의 핵심 블록체인 플랫폼인 하이퍼레저에 대한 관심이 늘어나면서 시무 프로젝트에 이를 적용하고자 하는 시도가 이루어지고 있다. 하지만 하이퍼레저 아키텍처에 대한 근본적인 이해 없는 단편적인 지식으로 인하여 어려움을 격고 있다. 본 튜토리얼에서는 대표적인 오픈소스 블록체인 플랫폼인 하이퍼레저 아키텍처를 소개하고 이를 이용한 응용 프로그램 개발의 핵심 요소를 알아본다. 이를 통해 하이퍼레저 응용프로그램을 개발 할 수 있는 능력을 키우고 실무에 적용할 수 있는 기반을 마련한다.
- 약력:
  - 단국대학교 소프트웨어학과 교수
  - 자율형 블록체인 융합연구소 소장
- 연구분야: 블록체인, 스마트 계약, 자율 플랫폼

튜토리얼 T4: SW 위험원 분석

- 일시: 1월 28일(월) 14:40~16:10
- 장소: 세미나실 1
- 제목: 소프트웨어 안전성을 위한 위험원 분석 기법
- 연사: 권기현 교수(경기대)
- 튜토리얼 초록:

  모든 사물이 소프트웨어로 제어되는 4 차산업시대에서, 소프트웨어 안전성은 아무리 강조해도 지나치지 않다. 소프트웨어 안전성을 위해서는 안전을 위협하는 위험원을 식별하고, 평가하고, 저감대책을 수립하는 일련의 위험원 분석 활동이 중요하다. 본 강연에서는 첫째, 위험원 분석에 전통적으로 사용되어온 고장 기반의 기법들을 소개한다. 특히, 기능 안전과 관련된 국제 표준 동향도 설명하고자 한다. 둘째, 고장에 기반한 전통적인 위험원 분석과는 달리, 소프트웨어의 사고 원인은 다양한 컴포넌트들의 상호작용에 있는 만큼, 구성요소들간의 제어를 중심으로 한 위험원 분석 기법을 소개한다. 셋째, 이들 위험원 분석 기법을 적용한 경험을 소개한다.
- 약력:
  - 1985 년 경기대학교 전자계산학과 학사
  - 1987 년 중앙대학교 전자계산학과 석사
  - 1991 년 중앙대학교 전자계산학과 박사
  - 1991~현재 경기대학교 컴퓨터공학과 교수
  - (전) 미국 카네기멜론대학교 전산학과 연구교수
  - (전) 한국정보과학회 소프트웨어공학 소사이어티 회장
  - (현) 과기정통부 정책자문단 위원
  - (현) 한국연구재단 전문위원
  - Functional Safety Engineer Certificate
  - Functional Safety Professional Certificate

- Functional Safety Expert Certificate

◆ 연구분야: 정형 명세, 정형 검증, 시스템 및 소프트웨어 안전성

튜토리얼 T5: 프로그램분석/결함 검출

◆ 일시: 1월 28일(월) 14:40~16:10

◆ 장소: 세미나실 2

◆ 제목: 프로그램 분석을 사용한 결함 검출 (Bug Detection via Program Analysis)

◆ 연사: 류석영 교수 (KAIST)

◆ 튜토리얼 초록:

프로그램 분석은 프로그램을 실제로 실행하지 않고 프로그램 안에 있는 결함을 검출할 수 있는 기법이다. 본 튜토리얼에서는 자바스크립트 웹 앱부터 안드로이드 앱을 포함하여, 최근에는 블록체인의 스마트 컨트랙트까지 다양한 프로그램에 존재하는 여러 종류의 결함을 검출하는 분석 기법과 분석 도구 개발에 대해 이야기한다.

◆ 약력:

- 2009 년~현재 KAIST 전산학부, 교수
- 2005 년~2009 년 Sun Microsystems, 연구원
- 2001 년~2005 년 Harvard University, 연구원
- 2001 년 KAIST, 전산학 박사

◆ 연구분야: 프로그래밍 언어, 프로그램 분석

튜토리얼 T6: 데이터 사이언스

◆ 일시: 1월 28일(월) 14:40~16:10

◆ 장소: 세미나실 3

◆ 제목: 데이터 사이언스와 소프트웨어 공학 -반도체 분야 중심으로

◆ 연사: 박찬진 상무 (SK Hynix Data Science 담당)

◆ 튜토리얼 초록:

최근 AI, Big Data 기술의 발전과 기업의 디지털 전환 트렌드에 따라 데이터 사이언스에 대한 Job 과 조직에 대한 관심이 커지고 있다. 이는 ML 기술이 흥미로운 하나의 사례나 일회성 보고에 그치지 않고 실제 제품 생산 프로세스에 임베드되는 운영 시스템화를 통해 지속적 가치를 만들어가고 있기 때문이다. 본 튜토리얼에서는 반도체 기업에서의 데이터 사이언스 조직 역할, 주요 분석과제, 엔지니어링 활동을 소개한다. 특히 분석 환경, ML App 개발, 리뷰 및 테스팅, 통합, 배포, 성능, 모니터링, ML 모델 업데이트, ML 프레임워크, 플랫폼, 서비스 등 ML 기술의 운영 시스템화를 위한 SE 기술의 현황과 발전 방향에 대해 논의한다

◆ 약력:

- 2017 년 ~ 현재 SK Hynix Data Science 담당 상무
- 2014 년 ~ 2016 년 서울대 차세대 융합기술원 책임 연구원
- 2006 년 ~ 2014 년 LG 전자 수석 연구원
- 1994 년 ~ 1998 년 LG 소프트웨어 주임 연구원
- 2006 년 서울대 전기 컴퓨터공학 박사 (소프트웨어 공학)
- 2000 년 서울대 전산학과 석사 (소프트웨어 공학)
- 1989 년 ~ 1994 년 서울대 계산통계학과 학사

◆ 연구분야: 소프트웨어 아키텍처, 역공학, 데이터 사이언스, 빅데이터 분석

KCSE 2019 기조연설

기조연설 I

- 일시: 1월 28일(월) 16:40~17:30
- 장소: 그랜드홀 2
- 제목: 연구원 사례 중심의 R&D(연구개발) 품질관리
- 연사: 정효택 박사(ETRI 품질 혁신실)

기조연설 II

- 일시: 1월 29일(화) 16:00~16:50
- 장소: 그랜드홀 2
- 제목: 4차 산업혁명과 소프트웨어 안전, 그리고 우리의 사명
- 연사: 민상윤 대표 (솔루션링크)

기조연설 III

- 일시: 1월 29일(화) 16:50~17:40
- 장소: 그랜드홀 2
- 제목: 사물 인터넷을 넘어 사물 경제 시대로의 변화
- 연사: 박수용 교수 (서강대학교)

KCSE 2019 신임교수 세미나

신임교수 세미나 1: Bug Detection Using Mining Software Repository

- 일시: 1월 28일(월) 17:30~18:20
- 장소: 세미나실 1
- 제목: 마이닝 소프트웨어 레포지토리를 이용한 버그검출도구 개선
- 연사: 남재창 교수 (한동대)

- 세미나 초록:

개발자들이 버그검출도구의 사용을 꺼리는 이유 중 하나는 높은 오탐지율 때문이다. 본 세미나에서는 GitHub와 같은 소프트웨어 레포지토리의 다수의 오픈소스 프로젝트들을 이용하여, 기존 버그검출패턴의 오탐지율을 낮추거나 오탐지율이 적은 새로운 버그패턴을 설계하는데 활용할 수 있는지를 살펴 본 사례연구에 대해 발표한다. 본 연구에서는 1,622개의 버그 패치를 분석하여 만든 10개의 버그검출패턴을 구현하였고, 1,880여개의 자바 오픈소스 프로젝트에 단계적으로 적용하여 오탐지율을 낮출 수 있도록 버그검출패턴을 반복적으로 개선하였다. 반복 및 검출결과 기반으로 버그패턴을 개선하는 이 과정을 FeeFin Process로 제안하였다. FeeFin을 이용해 57개의 새로운 버그를 찾아 오픈소스 프로젝트 개발자의 확인을 받았으며, 44개는 실제로 수정 반영 되었다.

- 약력:

2018-현재 : 한동대학교 전산전자공학부 조교수
2017-2018 : 포항공대 컴퓨터공학과 연구조교수
2015-2017 : Postdoctoral fellow, Dept. of Electrical and Computer Engineering, University of Waterloo, Canada
2009-2015 : Ph.D., Dept. of Computer Science and Engineering, The Hong Kong University of Science of Technology, Hong Kong, China

- 연구분야:

Software Quality Prediction, Mining Software Repositories, Empirical Software Engineering

신임교수 세미나 2: 소프트웨어 공수 예측 정확도 향상

- 일시: 1월 28일(월) 17:30~18:20
- 장소: 세미나실 2
- 제목: 데이터 분할법에 의한 소프트웨어 공수 예측 정확도 향상
- 연사: 서영석 교수 (영남대학교)

- 세미나 초록:

최소제곱회귀법(LSR)은 가장 일반적으로 사용되는 소프트웨어 공수 예측 기법들 중 하나이다. 그러나, LSR 기법을 통해 생성되는 공수 예측 모델(단일 LSR 모델)은 소프트웨어 프로젝트 데이터 셋의 분포에 큰 영향을 받는다. 즉, 좁은 분포를 가진 소프트웨어 프로젝트 데이터셋에서는 단일 LSR 모델이 정확한 공수 예측 값들을 제공할 수 있는 가능성이 높지만, 넓은 분포를 가진 소프트웨어 프로젝트 데이터 셋에 서는 단일 LSR 모델을 왜곡시킬 수 있는 영향치들(예를 들어, 이상치)을 많이 포함하고 있기 때문에 모델이 올바르게 생성되지 않아 정확한 공수 예측 값들을 제공할 수 있는 가능성이 낮아진다. 따라서 본 세미나에서는 기존에 제안된 기법들에 비해 보다 정확하고 안정적인 공수 예측 값을 제공하는 데이터 분할 기반의 다중 공수 예측 기법 모델을 소개한다. 또한 소프트웨어 프로젝트 데이터 셋에서 이상치가 단일 LSR 모델의 공수 예측 정확도에 미치는 영향에 대해 실험적으로 조사해 본 후 그 영향에 대해 함께 논의한다.

- 약력:

2016-현재: 영남대학교 컴퓨터공학과 조교수
2014-2016: 한국산업기술시험원(KTL) 선임연구원
2012-2013: KAIST 정보전자연구소 박사후연구원
2008-2012: Ph.D in Computer Science, KAIST

- 연구분야:

Software cost estimation, Mining software repository, System of systems, Software defect analysis, Software process improvement.

신임교수 세미나 3: 교차프로젝트 결함예측

- 일시: 1월 28일(월) 17:30~18:20
- 장소: 세미나실 3
- 제목: 클래스 불균형을 고려한 교차프로젝트 결함예측
- 연사: 류덕산 교수 (전북대학교)

- 튜토리얼 초록:

  소프트웨어 결함 예측은 오류 취약 모듈에 테스팅 자원의 최적화된 할당을 돕는다. 내부 데이터로 모델을 구축하는 내부 프로젝트 결함 예측 (WPDP)과 달리, 과거 데이터가 없는 경우, 외부 데이터를 이용하는 교차 프로젝트 결함 예측 (CPDP)이 유용하다. CPDP 의 난제는 트레이닝과 테스트 데이터간 분포차이며, 타겟과 유사한 소스 데이터로 모델을 구축할 필요가 있다. 결함 데이터는 클래스 불균형 문제 (결함 클래스 인스탄스 갯수가 비결함 클래스 대비 매우 적음)를 지니며, 이러한 특성은 모델 성능을 저하시킬 수 있다. 위 두가지 문제들을 동시에 해결하기 위한 기법을 소개한다. 과거 데이터가 없는 회사들이, 이 기법을 사용하여, 데이터가 충분히 수집될 때까지 소프트웨어 품질을 예측할 수 있게 된다.

- 약력:

  2018-현재: 전북대학교 소프트웨어공학과 조교수
  2016-2018: KAIST 연구조교수
  2012-2016: KAIST 박사과정

- 연구분야:

  소프트웨어 애널리틱스, 소프트웨어 신뢰성 공학

논문 목차

일반논문

제목	페이지#
휘처모델과 태스크 유사도 기반 추천 시스템을 이용한 R&D 문서 재사용 지원	1
기계학습 기반 자율 제어 시스템의 참조 아키텍처	10
OpenDS 를 이용한 자율주행소프트웨어 시뮬레이션	12
시나리오 테이블을 이용한 STPA 에서의 사고 원인 식별	18
크로스 플랫폼 애플리케이션의 테스트 자동화 방안	20
Engine Control Unit (ECU) 실시간 객체 지향 모델의 능동 객체 기반 우선 순위 전파 멀티 코어 멀티 태스킹	29
패혈증 진단용 REBA 검사 자동화를 위한 영상인식 모바일 어플리케이션 개발	39
C 언어로 작성된 OSEK / VDX OS 모델을 사용한 임베디드 제어 소프트웨어 검증	47
모델 기반 멀티타스크 프로그램의 API 호출 안전성 검증을 위한 오경보 식별 및 성능 향상 기법	55
코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 향상	57
RNN 을 이용한 이슈 분류	67
서비스 기반 V2X 소프트웨어 환경에서의 서비스 소비 품질 최적화 방법	76
캡슐내시경 영상 데이터 중심 소프트웨어 개발을 위한 데이터 셋 품질 평가 매트릭	78
스마트 컨트랙트 및 DApp 신뢰성 향상을 위한 단계별 체계적인 테스트 기법	88

단편논문

제목	페이지#
차량 수준 Hazard Analysis Risk Assessment 활동의 취약점과 실용적해결방안 연구	90
스마트 빌딩을 위한 b.IoT 솔루션과 휴먼센싱 기술	94
다양한 신경세포 모델의 행동 양식 특징 분류를 통한 효과적인 코드관리	96
대사증후군 예방을 위한 EMC 프로그램 개발	100
모놀리식 애플리케이션의 컨테이너 기반 마이크로서비스 전환을 위한 스크립트 생성 방법	103
GAS 절감이 가능한 스마트 컨트랙트 코드 패턴 분석 방법	105
솔리디티 스마트 컨트랙트의 스크립트 기반 시나리오 테스트	109

보안성을 고려한 경량 크래시 리포트 생성 기법 113
Generative Adversarial Network 을 이용한 소프트웨어 결함 예측 116
사이버-피지컬 시스템의 상호운용성 상충관계 분석에 관한 연구 121
어휘적 특징에 기반한 프로그램 자동 패치 검색 공간 축소 125
소프트웨어 제품 라인 개발을 위한 가변성 모델의 무모순성 검증 130
Automatic Data Flow Analysis to Generate Test Cases from Activity Diagrams 132
3D 시나리오 저작도구 개발 136
순환 인공 신경망을 이용한 코드 변경 추천 139
시스템 오브 시스템즈 협업 정책 구성을 위한 상향식 기법 제안 142
BIMM 에 기반한 소프트웨어 산업의 비즈니스 혁신성 가이드를 위한 적용사례 146
위치 분석 기반 통합 클라이언트-서버 프로그램 컴파일 방법 구현 149
선형 SVM 과 Random Forest 를 이용한 DNA 메틸화 영역 예측 153
순환신경망을 활용한 맥락 데이터 기반 사용자 인증 기법 157
MDA 기반 태양광 에너지 모니터링 시스템의 유지보수를 위한 비용 감소 방안 사례 161
위해도 분석 결과의 효과적인 확인을 위한 추적성 기반 위해도 모델 164
병행성 결함 탐지 도구에 대한 실험적 연구 168
항공기 프로펠러 고장진단을 위한 압력센서 비교 172
대인관계 개선 VR 콘텐츠 생성을 위한 에디터 구현 방법에 관한 연구 176
테스트케이스의 제어흐름기반 프로그래밍 실패 피드백 시스템 설계 180
버그 검출 규칙 개선을 위한 코드 문맥 수집 방법에 대한 연구 184
교차 프로젝트 기반 토픽 모델링과 네이브 베이지안 다항분포을 이용한 버그 리포트의 우선순위 예측 188
안전 필수 시스템의 ISO/IEC 16085 기반 위험 분석 방법 비교 192
Wright 표기법을 이용한 무인 자율주행 자동차 그룹 이탈 결과 시스템 분석 196

산업체논문

제목 페이지#
빅데이터 분석용 DBMS 를 위한 테스트 자동화 프레임워크 설계 200
Automatic Usage Profiling 을 통한 초기 앱 실행 속도 개선 방법 207
System Test 자동화 도구 제품화를 위한 품질 개선 209
애자일 전환을 위한 핵심요소 213
Testing_AI_Systems 232

고품질 무기체계 소프트웨어를 위한 신뢰성 시험 환경 구축에 관한 연구 247
무기체계 소프트웨어 신뢰성 통합관리시스템 소개 249
고신뢰성 내장형 무기체계 소프트웨어를 위한 실행 가능한 모델링 언어 265
차량용 ECU 소프트웨어의 모델 기반 개발 시 백투백 비교 테스트 방법 272

학부생논문

제목	페이지#
오픈소스 소프트웨어 저장소에서 결함 수정 코드 수집 방안	276
강화학습을 적용한 주행모드 변화에 따른 하이브리드 자동차 배터리 SOC 밸런싱 연구	280
최신 수면 데이터 분석 플랫폼 설계 및 구현	284
LoRa 기반의 치매환자를 위한 안전관리시스템	288
"시뮬레이션-이력 데이터 분석 서비스 프레임워크 설계 및 구현"	293
행정안전부 소프트웨어 보안 취약점 진단기준과 Java 웹 어플리케이션 대상 오픈소스 보안 결함 검출기 검출대상의 총체적 비교	298
중복된 이슈 보고서 요약	301
동의어 치환을 이용한 심층 신경망 모델의 테스트 데이터 생성	305
시장도매인 위탁 판매를 위한 블록체인 기반 시스템 설계	307
TF-IDF 알고리즘과 API 정보를 활용한 윈도우 실행파일 분류	311
의료 영상 데이터 중심 소프트웨어 개발을 위한 학습 노이즈 영상 분류 기법	317

휘처모델과 태스크 유사도 기반 추천 시스템을 이용한 R&D문서 재사용 지원

남승우 O, 혼다네스, 홍장의
충북대학교 컴퓨터과학과
{swnam, horndaneth}@selab.cbnu.ac.kr, jehong@chungbuk.ac.kr

Supporting R&D document reuse using feature model and task similarity - based Recommendation System

Seungwoo Nam O , Horn Daneth, Jang-Eui Hong
Department of Computer Science, Chungbuk National University

요 약

소프트웨어 재사용은 이미 입증되고 신뢰할 수 있는 소프트웨어 컴포넌트를 사용하여 새로운 소프트웨어 시스템을 개발하는 활동이다. 재사용은 요구사항 문서, 디자인 문서, 소스코드 등과 같은 모든 종류의 소프트웨어 산출물로 수행할 수 있다. 전통적인 재사용 접근법은 소프트웨어 개발을 위한 적절한 컴포넌트를 찾는 활동이기 때문에, 아직 소프트웨어 개발과 관련된 내용이 들어있지 않은 R&D(Research & Development)문서에 적용하기에는 한계가 있어 새로운 추적 기법이 필요하다. 그리고 R&D문서를 재사용하는 사용자는 연구내용의 유사성 외에 연구작업의 유사성 비교를 통한 폭넓은 연구방향 제시를 원하기 때문에 이에 대한 해결책도 필요하다. 본 논문에서는 사용자의 요구에 충족하는 R&D문서의 재사용을 위해 기존 소프트웨어 제품군에 맞춰진 SPL(Software Product Line)을 R&D문서 재사용에 적절한 형태로 확장한 RSPL(Research-SPL)을 제안하고, 휘처모델과 태스크 유사도를 기반으로 하는 R&D문서 재사용 추천 시스템 FTRS(Feature model and Task similarity - based Recommendation System)을 제안한다. 제안하는 시스템은 사용자가 원하는 R&D문서의 효율적인 재사용을 가능하게 한다.

1. 서 론

소프트웨어 재사용은 이미 입증되고 신뢰할 수 있는 소프트웨어 컴포넌트를 사용하여 새로운 소프트웨어 시스템을 개발하는 활동을 말하며, 비용 절감과 품질 보증 모두에 있어서 중요하다[1]. 모든 기업들은 높은 품질의 정보시스템과 높은 생산성을 가지는 소프트웨어 제품을 만들기 위해 소프트웨어 제품라인 공학(Software Product Line Engineering, SPLE)을 도입하고 있다[2].

R&D문서는 제품에 대한 설계 및 절차, 개발계획, 명세 내용이 들어있는 문서로써, 소프트웨어 제품을 개발하기 전에 작성되어야 한다. R&D문서를 재사용하는 것은 효율적인 소프트웨어 제품 개발을 가능하게 하는 방법으로, 최근에는 R&D(Research & Development)문서의 높은 생산성을 위해 문서의 섹션을 재사용 단위로 모듈화 한 연구와 R&D문서의 재사용을 지원하기 위한 프레임워크에 관한 연구가 있었다[3,4]. 연구에서 제안한 프레임워크는 많은 페이지 수와 다양한 목차로 구성되어 있는 R&D문서 중에서 필요한 부분만 효율적으로 재사용하기 위해 고안되었으며, 소프트웨어 컴포넌트에 대한 재사용 개념을 R&D문서에 적용하여 소프트웨어 컴포넌트의 기원이 되는 소프트웨어 연구 및 개발 산출물을 재사용할 수 있도록 하였다.

<그림 1>은 R&D문서의 재사용 단위인 mC(micro Component, 문서의 절을 모듈화한 재사용 단위)와 mC의 재사용을 위한 확장 패턴을 나타낸다[5].

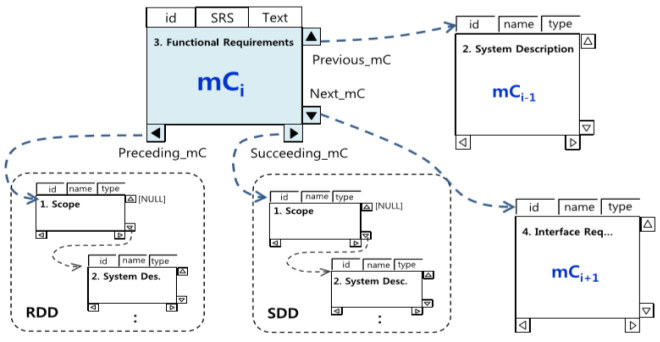


그림 1. Logical connections between mCs to provide traceability[5]

해당 연구에서 해당 프레임워크에서 의미적인 탐색을

지원하기 위하여 탐색 기준이 되는 mC_i는 식별자 이외에도 개발단계, 기술유형을 속성으로 가지고 있다. 사용자는 현재 자신이 작성중인 문서의 단계와 유형을 비교하면서 mC를 재사용할 수 있다. 확장 패턴 중에서 Preceding_mC와 Succeeding_mC는 다른 문서 혹은 다른 버전에 대한 탐색을 위해 제공되며, Previous_mC와 Next_mC는 같은 뿌리를 갖는 문서(e.g. 단일 문서 내의 1.1절과 2.2절은 같은 뿌리를 갖는다)의 구조에 대한 탐색이다.

검색 키워드와의 유사도를 통해 가장 높은 유사도 값을 가지는 후보 mC들을 결과로 추천해주는 시스템[6] 및 <그림 1>과 같이 최종적으로 재사용하기 위한 mC 목록을 선정하는 과정 중의 하나인 확장 패턴은 mC의 최종적인 재사용 결정 시간을 단축하기 위한 연구로, 사용자가 현재 수행중인 R&D 연구와 비슷한 mC를 최종적으로 재사용하기 위한 방안으로 제시되었다. 특히 키워드와의 유사도를 이용한 mC 추천시스템은 사용자의 현재 연구내용과 mC가 가지고 있는 연구내용의 유사성을 기반으로 mC 재사용 결정에 도움을 주지만, 선택한 mC 외의 추가적인 mC 재사용 결정에 큰 도움을 주지 못한다. 그래서 mC의 재사용성 향상을 위해 연구내용 유사성 외에 연구작업(research task)의 유사성을 기반으로 하는 추천시스템을 제안한다. 연구작업은 연구에 필요한 산출물을 선택해서 재사용하는 행위이다. 연구작업 유사성은 연구내용 유사성과 다르게 산출물의 내용을 고려하지 않는다. 연구작업 유사성을 재사용에 이용하기 위해 연구작업을 정의하고, 제안하는 추천시스템의 개념적 설명을 위한 SPL을 R&D문서에 적용하기 위해 확장한 RSPL을 제안한다.

SPL은 기본적으로 소프트웨어 컴포넌트 재사용을 기반으로 하며, Domain Engineering의 산출물을 개발하기 위하여, 또는 Application Engineering의 산출물을 개발하기 위하여 재사용이 시도된다. 제품에 대한 공통성(commonality)과 가변성(variability)을 기반으로 개발활동이 진행되지만, 개선된 시스템을 개발하거나 새로운 기술을 접목하는 제품 군에 대하여는 연구 개발 및 연구 탐색과정을 거쳐 이루어지는 것이 대부분이다. 이러한 상황이 본 연구에서 RSPL을 제시하고자 하는 이유이다.

본 논문 2장에서는 컴포넌트 재사용과 연관된 기존의 기법 및 연구들을 확인한다. 3장에서는 관련 연구에서 발견한 한계점들을 해결하기 위한 방법론을 제안한다. 4장에서는 제안한 방안들을 도구로 구현하고 마지막 5장에서는 본 논문에 대한 결론을 내린다.

2. 관련연구

먼저 소프트웨어의 재사용성을 위한 SPL에 관한 연구를 살펴보면, K. Pohl[7]은 SPLE 프레임워크를 정의하여 기존의 단일 시스템 개발과 SPLE을 이용한 개발의 차이점을 설명하였다. 제안한 프레임워크는 기존 시스템 개발에는 없는 2개의 Domain Engineering, Application Engineering 프로세스를 가진다. Domain Engineering은 도메인 산출물을 개발하는 것뿐만 아니라 제품 라인의 공통성과 가변성을 가지는 휘처(feature)를 식별하여 정의하고, Application Engineering은 도메인 산출물로부터 실제 어플리케이션을 만들어낸다. SPLE 패러다임은 향상된 생산성이라는 이점을 가지고 있어 소프트웨어 개발 및 관리에 자주 사용되지만, 기능 및 품질 요구사항이 확정되지 않은 R&D문서 개발에 적용하기에는 한계가 있다.

R. Mitschke[8]의 연구는 SPL에서 만들어지는 휘처 모델에 대한 버전 및 의존성들을 관리할 수 있는 구성관리 시스템을 제시하여 SPL의 포괄적인 추적성을 가능하게 하였다. D. Ko[9]는 SPL에서 만들어지는 소프트웨어 산출물과 그 산출물로부터 파생된 제품 간의 일관성을 유지하기 위해 자원에 대한 버전 식별자를 사용하고 예제 시스템에 적용하여 타당성을 입증했다. 위의 연구들은 기능 및 품질 요구사항을 기반으로 정의된 휘처가 사용자의 피드백에 따라 변할 수 있기 때문에 이러한 변화를 관리하기 위한 버전 관리 기법 및 휘처를 추적할 수 있는 기법을 제안하였다. 하지만, 단계별 R&D문서가 단계별 소프트웨어 산출물 자체를 의미하는 것이 아니기 때문에 R&D문서에 대한 버전 관리 및 추적에 위의 연구들을 그대로 적용하기는 힘들다.

W. Maalej[10]은 사용자와 툴(tool)에서 작성되는 산출물간의 상호작용(e.g. open, edit)을 하나의 context로 정의하고 사용자의 태스크에서 상호작용이 빈번한 산출물들이 다른 태스크에서도 사용된다면, 두 태스크는 관련이 있다고 가정했다. Jaccard indexing에 사용될 관련성 모델을 정의하고 해당 모델을 이용한 Jaccard 값이 높을수록 context가 유사하다고 판단했다. 산출물의 내용 유사도를 기반(latent semantic indexing based)으로 하는 추천시스템이 관련이 있는 두 태스크를 잘 식별하고 추천해주는 반면에, 이미 관련이 있거나 관련이 없는 태스크를 식별하는 데는 어려움이 있어 이를 보완하기 위한 기법으로 Context-similarity를 사용하였다. 위의 연구는 산출물을 하나의 재사용 단위로 보지 않고 재사용을 지원하기 위한 방안을 고려하지 않았기 때문에, 이를 R&D문서 재사용 추천 시스템에 적용하기에는 다소 미흡한 점이 있다.

관련 연구들의 분석을 통해 SPL, Content-similarity, Context-similarity와 같이 재사용을 지원할 수 있는 기법들이 존재하지만, 연구작업의 특성을 고려한 재사용 지원에 활용되지 않았음을 확인하였다.

3. R&D 기반 재사용 지원 프레임워크

본 논문의 연구수행 범위는 <그림 2>와 같다. 사용자는 R&D문서를 작성하는 도중에 연구에 필요한 내용을

현재 위치에 재사용하기 위해 임포트 요청을 하고, 적절한 키워드를 이용하여 mC를 검색한다. 검색 키워드와 가장 관련이 있는 순서대로 후보 mC들이 결과로 나타나면, 사용자는 현재 자신이 수행중인 연구와 가장 관련이 있는 mC하나를 선택하게 된다. 추가적인 mC를 재사용하기 위해 R&D문서를 확장 패턴을 이용하여 탐색하면서 재사용할 mC를 목록에 추가한다. 재사용할 mC를 목록에 추가하는 도중에 본 논문에서 제안하는 재사용 추천 시스템을 통해 사용자와 연구가 유사한 태스크를 확인하고, 해당 태스크에 사용된 mC의 내용을 확인하여 목록에 추가할 것인지를 결정한다. 최종적으로 재사용할 mC 목록을 선정하고 작성중인 문서로 임포트한다.

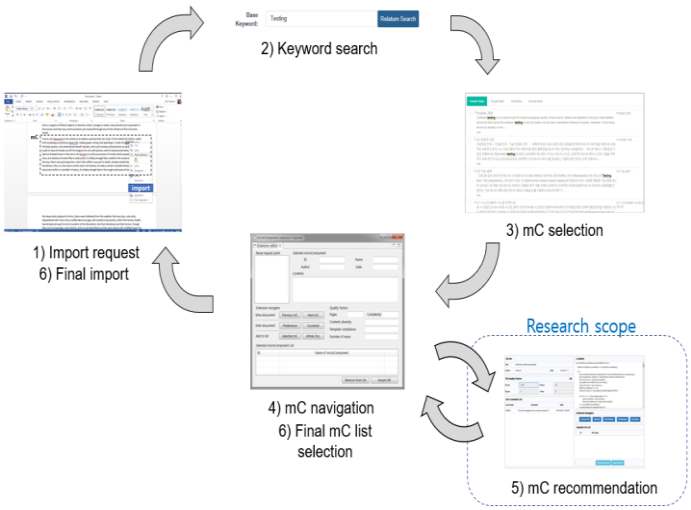


그림 2. Operational concept of our reuse framework

3.1 mC 재사용 휘처 모델과 RSPL

먼저, 본 논문에서 제안하는 R&D문서 재사용 추천 시스템에서 사용하는 mC의 휘처 모델을 정의하여 휘처의 추적 방안을 제시하고자 한다. <그림 3>은 mC에 대한 휘처 모델과 휘처 모델로부터 파생된 mC 인스턴스를 나타낸다. mC는 R&D문서를 섹션별로 재사용한 단위이기 때문에 mC마다 R&D문서의 Field와 Engineering 휘처를 상속받는다. 예를 들어, 자동차에 탑재되는 임베디드 소프트웨어에 대한 요구사항 식별 및 명세에 관한 R&D문서이면, 해당 R&D문서에 속하는 모든 mC들은 Automotive와 Requirement 휘처를 갖는다. mC마다 개발과 직접적인 내용을 가지거나 가지지 않을 수 있기 때문에 Development 휘처는 선택사항이다.

<그림 4>는 기존 SPL 개발 프로세스를 R&D문서 개발을 위해 확장하여 정의한 RSPL을 나타난다. RSPL을 구성하는 단계는 다음과 같다.

(1) Research Feature Modeling은 mC의 메타정보로 활용될 수 있는 휘처들을 모델링하는 단계로 <그림 3>의 휘처들이 이 단계에서 모델링 된다.

(2) Research Domain Design은 mC의 재사용을 위한 핵심 개념들을 명세하는 단계이다. 이 단계에서 mC가 Development 휘처를 가지고 있으면 Development Feature Modeling 단계에 있는 휘처를 추적할 수 있다.

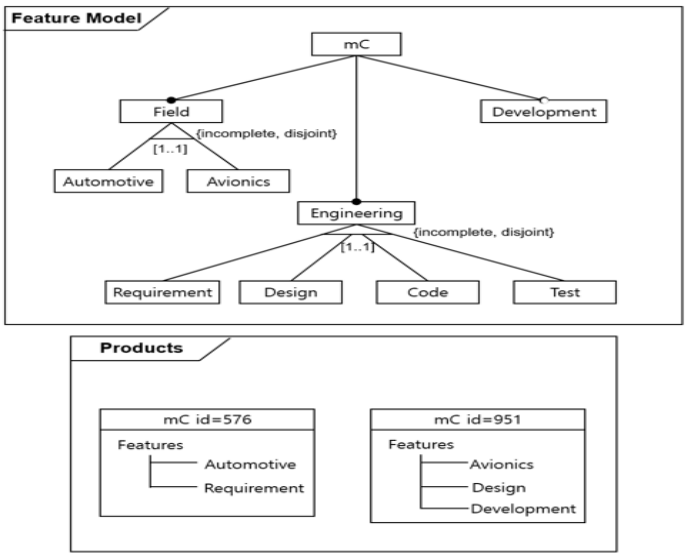


그림 3. Feature model and its instance of the mC

(3) Development Feature Modeling은 연구내용을 실현하기 위한 개발에 대한 요구사항(e.g. 기능 및 품질 요구사항)을 휘처 모델로 정의한다.

(4) Development Domain Design은 아키텍처 레벨에서 SPL의 실현을 위한 핵심 개념들을 명세하는 단계이다.

(5) Development Domain Realization은 이전 단계의 컴포넌트 모델의 구현을 위해 구체적인 분석을 수행하는 단계이다.

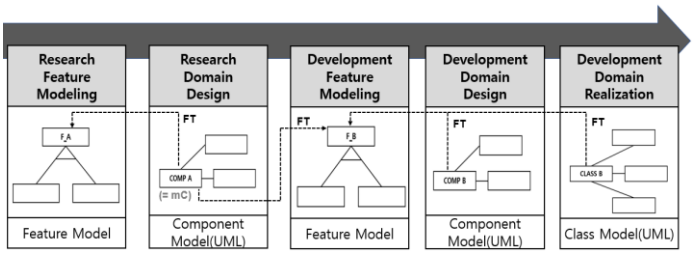


그림 4. RSPL Development Process and Artifacts

<그림 5>는 <그림 3>의 휘처들을 mC의 메타정보로 사용한 것을 나타낸다. 기본적인 mC의 id, title 외에 재사용을 위한 휘처인 “field”와 “engineering” 속성이 포함되어 있는 것을 알 수 있다. 그림에서 단락 구분을 통해 id=1이 최상위 노드인 것을 나타내며, 2와 4은 1의 자식 노드, 3은 2의 자식 노드, 5는 4의 자식 노드인 것

을 알 수 있다.

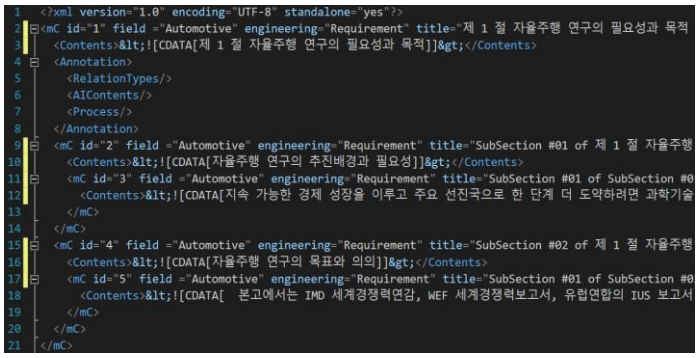


그림 5. An example of features used as attributes

<그림 3~5>를 토대로 재사용을 위한 mC의 휘처 모델링, 휘처 추적, 메타정보로써의 휘처 사용을 알 수 있었다.

3.2 태스크 유사도 기반 추천

3.1절에서 설명한 RSPL은 재사용된 mC의 유효성을 추적하고, 해당 mC의 내용을 기반으로 개발 산출물간의 추적도 제공한다. 태스크 유사도 기반 mC 추천 시스템 상에서, RSPL에서 생성된 메타모델은 사용자 연구에 적절한 mC를 추천하기 위해 사용 된다. 보통 사용자가 특정 mC에서 Import request를 요청할 때, 사용자가 원하는 재사용 후보 mC들은 Import request를 요청한 mC와 같은 휘처를 가진다. 하지만, 사용자가 재사용을 원하는 mC가 Import request를 요청한 mC와 다른 휘처를 가질 수 있다. 예를 들어, 모든 산업분야에 적용 가능한 전기/전자/프로그램 가능한 전자 안전 관리 시스템의 기능 안전 국제표준인 IEC 61508[11]은 <그림 3>의 Automotive 휘처를 가지는 mC뿐만 아니라 Avionics 휘처를 가지는 mC에서도 관련 내용이 사용될 수 있다. 이때 Automotive 휘처를 가지는 mC에서 위의 내용에 해당하는 Avionics 휘처를 가지는 mC를 재사용하려고 하면 휘처가 상충되게 된다. 따라서, 이를 해결하기 위해 각 mC의 휘처를 고려하는 것이 아닌 RT(research task)의 휘처를 고려한 재사용 추천을 제안한다. 즉, RT와 RT에 속하는 mC가 다른 휘처를 가질 수 있으며, 같은 휘처를 가지는 RT의 유사도를 기반으로 다른 사용자에게 해당 RT와 다른 휘처를 가지는 mC를 추천하기 때문에, 의미가 없는 재사용 추천이 발생하지 않는다.

본 논문에서는 RT를 R&D문서를 개발하기 위한 일련의 작업으로 정의한다. 예를 들어 사용자는 특정 대상(주제)을 연구하고 연구 내용을 문서의 서론, 본론, 결론 구성에 맞게 작성하는 일을 할 때, RT를 수행한 것이다. mC는 기존의 (R&D) 문서를 절 단위로 구분한 것이기 때문에 mC의 내용을 작성하는 일도 연구 태스크를 수행한 것이라 할 수 있다. 또한, mC를 재사용하기 위해 목록에 추가하는 것도 사용자의 연구과정에서 나온 선택이기 때문에 RT를 수행한 것이다. 본 논문에서는 RT를 재사용을 위해 mC를 목록에 추가 및 임포트한 mC를 작성중인 문서의 흐름에 맞게 수정하는 것으로 한정하여, 제안하는 재사용 추천 시스템에 활용한다.

<그림 6>은 FTRS(Feature model and Task similarity - based Recommendation System)에서 RT 간의 유사도 비교를 통해 사용자에게 후보 mC를 추천해주는 과정을 나타낸다. 소괄호는 시나리오의 순서를 의미한다. 첫 번째 시나리오가 진행되기 전에 사용자는 <그림 2>의 1)에서 4)까지의 과정을 거친다. 현재 재사용할 mC 목록을 유지하는 것을 RT_0라 할 때, RT_0의 휘처는 현재 사용자가 작성중인 R&D문서의 휘처인 Field, Engineering과 동일하다. <그림 6>에서 알파벳으로 표기된 사각형(e.g. A, B, C, D, E)은 RT에 속하는 mC를 의미한다. <그림 6>에서 Import request한 R&D문서가 자동차에 탑재되는 임베디드 소프트웨어에 대한 요구사항 명세(Software Requirement Specification)이기 때문에, Automotive와 Requirement 휘처를 가진다. 1) RT recommendation-1은 RT_0와 같은 휘처를 가지면서 태스크 유사도가 높은 후보 RT들을 추천해준다.

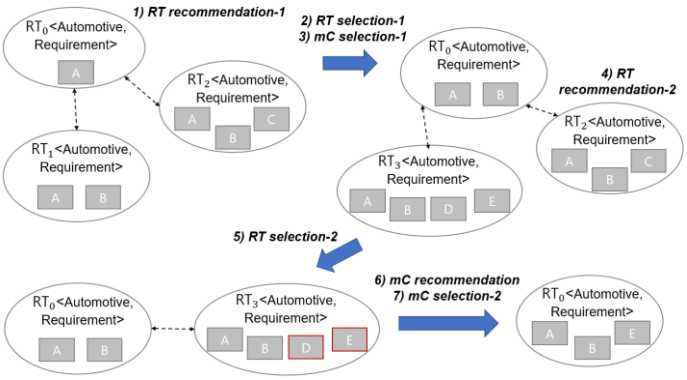


그림 6. An mC selection scenario on FTRS

태스크 유사도를 산출하기 위해 식(1)과 같은 Jaccard indexing[12]을 사용한다. 식(1)에서 RT_i와 RT_j의 합집합은 RT_i에 속하는 mC들과 RT_j에 속하는 mC들의 중복이 없는 합을 의미하며, RT_i와 RT_j의 교집합은 RT_i에 속하는 mC들과 RT_j에 속하는 mC들 중 공통으로 가지고 있는 mC들을 의미한다.

$$J(RT_i, RT_j) = \frac{|RT_i \cap RT_j|}{|RT_i \cup RT_j|} \quad -(1)$$

Jaccard indexing의 결과 RT_1이 RT_2보다 RT_0와 높은 태스크 유사도를 보이기 때문에 RT_2보다 높은 순위의 추천 목록에 위치한다($J(RT_0, RT_1) = 1/2$, $J(RT_0, RT_2) =$

1/3). 사용자는 두 개의 RT중에서 RT_1가 현재 자신이 구상하는 연구와 맞다고 판단하고 RT_1가 가지고 있는 mC를 재사용 목록에 추가한다. 2)번과 3)번 시나리오의 수행이 여기에 해당한다. 3)번 시나리오가 끝나게 되면 RT_0는 A와 B를 가지는 상태로 유지하게 된다. 4) RT recommendation-2 시나리오 실행 시, FTRS는 현재 RT_0와의 유사도 비교를 통해 후보들을 선택하고 목록으로 보여준다. 5) RT selection-2 시나리오는 RT_2가 RT_3보다 RT_0과의 태스크가 유사함에도 불구하고, 사용자는 C가 현재 구상하고 있는 연구와 거리가 멀다고 생각하여 RT_3을 선택하는 과정을 나타낸다. 5)번 시나리오가 끝나면 FTRS는 사용자에게 D와 E중 어느 것을 높은 순위로 추천할지 결정해야 한다. 순위 결정에는 식(2)와 같은 관계성(Relevance) 모델을 사용한다.

$$Rel(mC_i, RT_j) = \sum_{e \in IN(mC_i, RT_j)} \frac{Age(mC_i, RT_j)}{Freq(mC_i, RT_j)} \quad -(2)$$

*Rel(mCi,RTj)*는 특정 mCi가 해당 RTj에 얼마나 관련성이 있는지를 나타낸다. 최종적으로 임포트된 mCi가 최근까지 수정이력이 있고 수정횟수가 빈번하면 상대적으로 RTj에 관련이 없다고 판단된다.

$$Age(mC_i, RT_j) = \frac{max_{e \in IN(mC_i, RT_j)}(Sec_e)}{A} \quad -(3)$$

식 (3)의 *Age(mCi,RTj)*는 mCi에 작용한 마지막 이벤트 이래로 초 단위로 흘러간 시간의 정규화 함수를 의미한다. *IN(mCi, RTj)*는 RTj에 속하는 mCi와 상호작용한 모든 이벤트의 집합을 의미한다. 이벤트의 종류로는 임포트한 mCi를 수정거나 mCi에서 Import request를 요청하여 추가 임포트를 수행하는 두 가지 경우가 있다. 이 경우들은 임포트한 mCi를 원본 그대로 재사용할 수 없어 사용자가 많은 부분을 직접 수정했거나 또다른 임포트과정을 수행했다는 것을 암시한다. 여기에서 직접 수정은 mCi에서 다른 mC의 수정이 일어나기 전까지의 모든 수정 이벤트를 하나로 계산한다. *e*는 IN(mCi, RTj)에 속하는 하나의 인스턴스 이벤트를 의미하며, *Sece*는 이벤트 동작 후 흘러간 초 단위의 시간을 의미한다. *A*는 RTj에 속하는 mC들 각자 가장 최근의 이벤트 이후 흐른 시간들의 총 합을 나타내며, 정규화를 위해 사용되었다.

$$Freq(mC_i, RT_j) = \frac{\sum_{e \in IN(mC_i, RT_j)} * 1}{F} \quad -(4)$$

식 (4)는 *Freq(mCi, RTj)*는 RTj에 속하는 mCi에 작용한 모든 이벤트의 빈도수를 정규화 한 함수를 의미한다. F는 RTj의 모든 work session에서 발생한 총 이벤트 수를 의미한다. 여기서 work session은 RT에서 일어나는 모든 작업의 흐름을 이야기한다. 예를 들어, <그림 6>의 RT_1에 속하는 mC는 A와 B이며, A와 B에 대한 상호작용은 RT_1의 부분적인 work session이다.

FTRS는 6)번 시나리오에서 사용자에게 후보 mC를 추천해주고 사용자는 후보 mC 중에서 적절한 mC를 선택한다. 모든 선택이 끝나고 최종적으로 임포트가 되면, RT_0는 기존의 RT_1, RT_2, RT_3과 같이 저장소에 저장되어 다른 곳의 태스크 유사도에 쓰이게 된다.

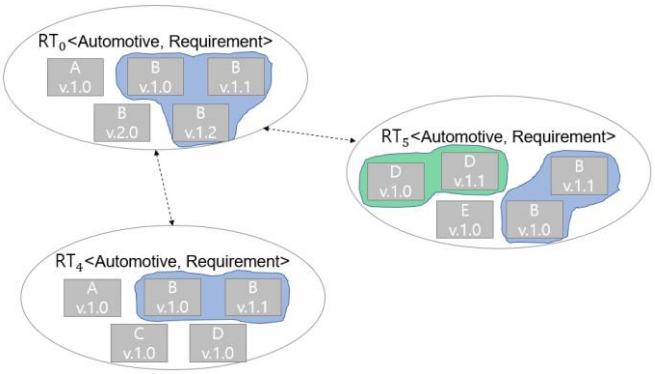


그림 7. A task similarity comparison considering version

재사용 목록에 추가되는 mC들은 같은 문서의 다른 버전일 수도 있다. <그림 7>은 mC들이 버전을 가지고 있을 경우를 보여준다. 현재 재사용 목록 상태를 나타내는 RT_0에서, 버전을 고려하지 않을 시에 A와 B를 재사용 목록으로 가지고 있는 것을 알 수 있다. 하지만 mC는 지속적인 변경을 하기 때문에 일관성을 위해 버전 관리를 해야 한다. 그래서 다음과 같은 버전 규칙을 따르도록 했다.

- Release Number: 1로 시작해서 전체를 뒤엎을 정도의 큰 변화가 발생했을 때 이 수치를 올린다.
- Major Number: 0으로 시작해서 주요 기능의 추가나 변경 등, 사용상 혹은 컨텐츠의 주요 변화가 발생했을 때 1 혹은 무작위로 증가한다.

Release Number는 mC의 저자가 직접 해당 mC를 수정했을 때 수치가 증가되는 반면에, Major Number는 사용자가 임포트한 mC를 직접 수정 혹은 해당 mC 상에서 또 다른 임포트를 하여 수정할 때 자동으로 수치가 증가된다. 예를 들어, <그림 7>에서 B(v.1.2)은 B(v.1.0)에서 2번의 수정 혹은 임포트 이벤트가 있었다는 것을 알 수 있다. B(v.1.x)는 변화가 일어난 mC의 뿌리가 B(v.1.0)으로 같기 때문에 같은 그룹으로 묶을 수 있지만, B(v.2.0)은 mC의 뿌리가 B(v.1.x)와 다르기 때문에 같은 그룹으로 묶을 수 없다. 이를 고려한 Jaccard indexing은 식 (5)와 같다.

$$\alpha J(RT_k, RT_l) = \frac{|RT_k \cap RT_l|}{|RT_k \cup RT_l|} + \frac{1}{\alpha} i \quad -(5)$$

제안하는 $\alpha J(RT_k, RT_l)$은 버전이 존재하는 mC들이 속하는 RT들의 태스크 유사도를 비교하기 위한 α-Jaccard indexing이다. 여기서 $1/\alpha * i$ 는 비교하는 RT 대상들이 실수부에서 같은 값이 나왔을 때, 1이상의 Major 버전을 가지고 있는 동일 mC가 많이 겹칠수록 유사하지 않은 태스크라는 것을 판별하기 위한 허수부이다. 왜냐하면 1이상의 Major 버전을 가지는 동일 mC가 많다는 것은 해당 RT에 사용하기 위해 많은 변경을 했다는 것을 의미하기 때문이다. RT_0, RT_4, RT_5의 B(v.1.0)은 동일한 내용을 가지고 있는 같은 mC이지만, B(v.1.1)은 다른 내용을 가지고 있는 서로 다른 mC이다. 이와 같은 버전을 태스크 유사도에 고려하기 위해 백과사전식 순서를 적용하였다.

백과사전식 순서는 여러 개의 부분 순서 집합들의 곱집합 위에 존재하는 부분순서이다[13]. 복소수 C 상의 순서관계 "<"를 a+bi, c+di에 대하여 식 (6)과 같이 두면 복소수 C는 순서집합이다. 복소수 C는 백과사전식 순서에 의한 전순서집합이 된다.

$$a + bi < c + di \Leftrightarrow (a < c)\ or\ (a = c\ and\ b < d) \quad -(6)$$

<그림 7>에서 α-Jaccard indexing을 이용한 RT_0에 대한 태스크 유사도 비교는 αJ(RT0, RT4)=2/5+1i, αJ(RT0, RT5)=1/5+1i로 RT_4와 더 유사하다. 사용자가 추천 RT 목록에서 RT_5를 선택했다면 추천 mC 목록을 구성하기 위해 D와 E 중 누가 RT_5와 가장 연관이 있는지 관련성 모델로 판별할 것이다. 두 개의 관련성 모델에서 D(v.1.0)과 D(v.1.1)에서 일어나는 이벤트는 하나의 D에서 일어나는 것과 같이 여긴다. 그림에서 E(v.1.0)은 임포트되고 수정이 이루어지지 않은 것을 의미하므로 이 때는 관련성 모델을 이용하지 않아도 추천 mC 목록을 생성할 수 있다. 순서는 E(v.1.0), D(v.1.1), D(v.1.0) 순으로 된다. D(v.1.1)가 D(v.1.0)보다 먼저인 이유는 D(v.1.0)을 RT_5에 그대로 사용할 수 없어서 수정했기 때문에 상대적으로 연관이 덜하기 때문이다.

<그림 8>은 버전업 된 mC가 해당 RT와 R&D문서 상에 반영되는 과정을 나타낸다. 먼저 사용자가 B(v.1.0)에서 임포트 과정을 거치게 되면, 새로운 RT_6이 생성된다. 이 태스크는 B(v.1.0)의 work session에 결박되어 임포트한 F와 G에 대한 이벤트가 발생하고 변경이 일어났을 시에 변경사항을 유지한다. 4)번 시나리오는 G(v.2.0)을 RT_6에 원본 그대로 사용하기 적절하지 않아 Import request를 수행한 것을 나타내며, 5)번 시나리오를 통해 새로운 RT_{n+1}을 생성한다. 생성된 RT_{n+1}은 RT_6과는 독립적인 태스크이다. G(v.2.0) 상에서 mC 임포트 및 수정이 이루어지면, RT_6에서 G(v.2.1)가 추가되고, 문서 상에서는 G(v.2.0)이 G(v.2.1)로 버전업이 된다.

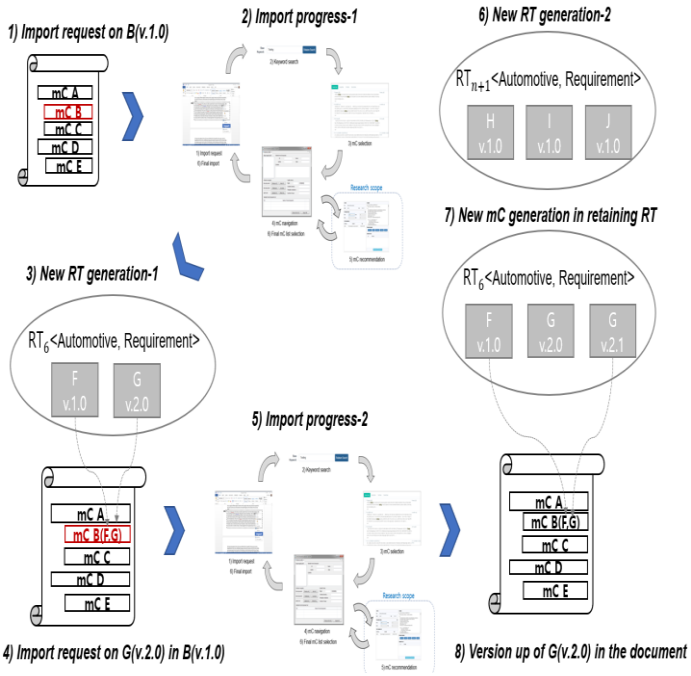


그림 8. Reflection of version up of the mC in the RT and document

4. 사례 연구

사례연구에서는 사용자와 FTRS의 상호작용을 위한 FTRS interface을 구현하고, FTRS 인터페이스를 이용한 사용 시나리오를 기술한다.

4.1 FTRS 인터페이스

FTRS 인터페이스는 다음과 같은 5개의 파트로 이루어져 있다; (1) *mC info*는 사용자가 특정 mC를 선택했을 때 그 mC에 대한 기본 정보를 보여주는 파트로, 기본 정보에는 mC의 제목, 버전, 저자, 생성된 날짜가 있다. (2) *Contents*는 사용자가 선택한 mC의 내용을 보여주는 파트이다. 사용자는 선택한 mC의 Contents를 보고 현재 수행중인 연구와 맞는지 판단한다. (3) *Reuse mC List*는 사용자가 재사용을 하기 위해 목록에 담은 mC들을 보여주는 파트로, 목록에서 특정 mC를 제거할 수 있는 *Remove from List* 버튼과 태스크 유사도를 비교해 RT를 추천해주는 *RT Recommend* 버튼, 그리고 FTRS interface를 종료하는 *Confirm* 버튼을 가지고 있다. (4) *Recommended RT List*는 태스크 유사도가 높은 순으로 RT를 정렬해 사용자에게 추천해주는 파트이다. (5) *Recommended mC List*는 사용자가 선택한 RT에 속하는 mC들 중에서 해당 RT와 가장 높은 관련성을 지니

는 mC들을 선순위로 추천해주는 파트이다.

<그림 9>는 FTRS의 인터페이스 예시를 보여주며, 그림에 나타난 화면은 사용자가 3개의 mC들을 Reuse mC List에 유지하는 중에 RT recommend 버튼을 눌렀을 때 나타나는 결과이다. 사용자는 FTRS가 추천해준 RT 목록 중 가장 첫번째를 선택했으며, 해당 RT에 속하는 mC들 중 첫번째 mC를 선택하였다. 선택한 mC에 대한 기본정보는 mC Info에 내용은 Contents에 나타난 것을 알 수 있다.

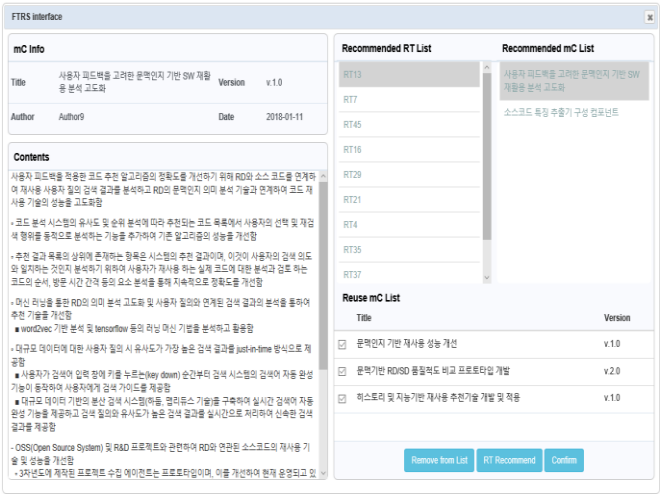


그림 9. An example of FTRS interface

4.2 FTRS 사용 시나리오

FTRS 사용 시나리오는 사용자 입장에서 보이는 FTRS의 역할을 잘 보여준다. FTRS 사용 시나리오는 다음과 같다. 시나리오의 단계 2에서, 사용자의 생성될 문서에 대한 메타정보 설정은 RSPL 상의 재사용 mC 에 대한 추적을 가능하게 한다.

[Scenario Name] FTRS를 이용한 소프트웨어 요구사항 명세서(SRS) 개발
[Scenario ID] FTRS-SRS-001
[Primary Actor] 요구사항 공학자
[Related Stakeholder] 프로젝트 매니저, 의뢰인, 설계 공학자, 품질 관리자
[Brief Description] 이 시나리오는 '문맥인지기반 SW 재활용 도구'의 SRS를 생성하는데 mC들을 어떻게 재사용하는지 묘사했다.
[Triggering Event] SRS 생성 요청
[Related Documents] 기술 보고서, 요구사항 정의서, 유사한 다른 프로젝트로부터 나오는 SRS
[Pre-Condition] 모든 관련된 문서들은 mC의 형태로 저장소에 저장되어 있다. 또한, 재사용 임포트의 이력이 있어 많은 RT와 여러 버전의 mC들도 저장소에 저장되어 있다.
[Post-Condition] 결과로 오는 SRS는 FTRS를 통해 추천되는 mC들을 이용하여 생성된다.
[Normal Flow of Scenario]
1. 사용자는 SRS R&D문서 템플릿을 선택한다.
2. 사용자는 생성될 문서의 메타정보로 Field: Reuse, Engineering: Requirement를 카테고리에서 선택한다.
3. 사용자는 생성될 문서에 기본정보를 입력한다 (i.e., 저자, 문서의 제목, 프로젝트의 제목 등).
4. 사용자는 특정 mC에서 다른 mC들을 재사용하기 위해 Import request를 한다.
5. 사용자는 특정 키워드를 통해 검색된 mC들을 확인한다.
6. 사용자는 현재 수행중인 연구에 적합한 mC 하나를 선택한다.
7. 사용자는 자신이 현재 진행하고 있는 연구와 부분적인 비슷한 흐름을 가지는 연구들이 있는지 확인하기를 원한다.
8. 사용자는 *FTRS interface*을 호출한다.
9. 사용자가 *RT Recommend* 버튼을 누르면, FTRS는 현재 사용자의 *Reuse mC List*에 담긴 mC들을 이용하여 RT 목록을 추천해준다.
10. 사용자는 *Recommended RT List*에 나타난 mC들 중에서 하나를 선택한다.
11. FTRS는 사용자가 선택한 RT에 속하는 mC 목록을 추천해준다.
12. 사용자는 *Recommended mC List* 중에서 자신의 연구와 관련이 있다고 생각되는 mC를 Reuse mC List에 추가한다.
13. 사용자는 FTRS interface을 그만 이용하고 싶어서, *Confirm*버튼을 눌러 확장 패턴 탐색이 있는 Navigation으로 돌아온다.
14. 사용자는 재사용 mC들을 최종 임포트한다.
15. 사용자는 SRS를 이어서 작성한다.

[Alternative Flow]
5a.1. 사용자는 키워드에 대한 결과가 마음에 들지 않으면 새로운 키워드로 검색한다.
5a.2. 사용자는 새로운 키워드에 대한 결과에 만족한다.
5a.3. 단계 6으로 간다.
7a.1. 사용자는 직접 mC들을 탐색하기를 원한다.
7a.2. 사용자는 Navigation의 확장 패턴을 통해서 문서 내 혹은 다른 문서의 mC들을 탐색한다.
7a.3. 사용자는 재사용할 mC들을 직접 탐색하는 것보다 자동으로 추천해주기를 원한다.
7a.4. 단계 7로 간다.
12a.1. 사용자는 추천 mC 목록 중에 수행중인 연구와 관련 있는 mC가 여러 개 있으면 해당하는

mC를 모두 재사용 mC 목록에 추가한다.
13a.1. 단계 9로 간다.
15a.1. SRS 작성 도중에 현재 커서가 위치해 있는 mC에서 새로운 mC를 임포트하기를 원한다.
15a.2. 단계 4로 간다.

4.3 사용 시나리오 재사용 분석

4.2절에서 제시한 사용 시나리오에 대한 재사용성 분석을 RT 특성을 고려하지 않은 키워드간의 유사성을 기반으로 검색하는 경우와 비교하면 다음과 같다. 이때 비교는 작성 문서의 절(즉, mC) 단위 재사용이 이루어진다고 가정한다.

<가정>
1) 키워드 간의 유사성 기반 검색과 FTRS는 공동의 문서 메타정보(e.g. id, title, version, author, date, contents, field, engineering, etc.)를 이용한다.
2) 키워드 간의 유사성 기반 검색과 FTRS는 같은 유사도 측정기법(e.g. Jaccard indexing)을 사용한다.
3) FTRS를 이용하지 않고 재사용 mC를 선택하는 사용자는 FTRS 사용 시나리오 상의 사용자와 동일하다.
4) 사용자는 가정 3)의 두 시나리오 상에서 모두 같은 대상에 관한 재사용 mC를 찾는다.

<재사용성>
- 키워드와 관련 없는 mC의 재사용 향상: FTRS는 태스크 기반의 추천 시스템이기 때문에, 특정 mC가 검색 키워드와 크게 관련이 없어도 사용자의 재사용 선택에 따라 추천 목록으로 올라갈 수 있다. 즉, 기존 다수의 사용자들이 서로 비슷한 성향을 가지는 연구를 수행하기 위해 특정 mC를 재사용에 이용할 경우, 해당 mC는 위의 사용자들과 비슷한 연구를 수행하는 새로운 사용자에게 필요한 mC일 수 있다. 키워드 기반의 검색일 경우에는 이를 고려하지 않지만 FTRS는 이를 고려하여 키워드와 관련이 없는 mC의 재사용을 향상시킨다.
- 사용자의 변경이력 제공: FTRS는 mC 저자에 의한 변경(using release version)과 재사용 mC 사용자에 의한 변경(using major version)을 이용해 지속적으로 버전을 관리한다. 특히, 사용자에 의한 변경된 mC는 연관된 RT에서만 관리하기 때문에 키워드를 통해 검색되지 않는다. FTRS는 기존 사용자가 선택한 재사용 mC를 변경한 이력까지도 새로운 사용자에게 제공하여 재사용성을 향상시킨다.
- 저자의 mC변경 추적제공: FTRS는 RSPL의 휘처 모델을 사용한다. 저자가 기존의 mC에 개발과 관련된 내용을 넣기 위해 Development 휘처를 추가로 선택하고 내용을 작성하면, 이 mC를 재사용한 사용자들은 mC에 대한 저자의 내용 변경뿐만 아니라 개발에 관련된 asset에 대한 추적 정보도 알림을 통해 제공받을 수 있다.

5. 결 론

재사용을 위한 방법론 및 추천 시스템에 대한 연구가 진행되고 있지만, 재사용 대상의 초점이 연구가 아닌 개발에 한정되어 있었다. 특히 R&D문서에 아직 개발에 관한 내용이 존재하지 않을 경우, 해당 R&D문서를 재사용하기 어려웠다. 한편, R&D문서의 재사용을 원하는 사용자는 연구내용의 유사성 외에 연구작업(Research Task)의 유사성까지도 비교하여 폭넓은 연구방향 제시를 원하기도 한다. 이를 위해, 본 논문에서는 R&D문서를 구성하는 mC의 재사용 향상을 위해 SPL을 확장한 RSPL를 제안하고 이의 휘처 모델을 정의하였다. 또, 정의한 휘처 모델과 태스크 유사도를 기반으로 하는 R&D문서 재사용 추천 시스템 FTRS를 제안하였다.

향후 연구로는 R&D문서를 재사용하기를 원하는 사용자가 FTRS를 사용했을 때와 사용하지 않았을 때, 최종적인 임포트까지 걸리는 시간과 재사용 만족도를 비교하는 연구를 수행하여 재사용 극대화를 위한 방안을 수립할 예정이다.

Acknowledgement
This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(NRF-2014M3C4A7030505).

6. 참고문헌

[1] D. K. Kim, et al., "Special issue on software reuse," Software Practice and Experience, Vol. 47, No. 7, pp.941-942, May 2017.
[2] S. Kim and J. A. Kim, "Consistency Checking Rules of Variability between Feature Model and Elements in Software Product Lines," Proceedings of KIIT Summer Conference, pp.519-524, Nov 2012.
[3] S. Kim, et al., "Toward Offline Contents Based Software R&D Support System," Advances in Computer Science and Ubiquitous Computing, pp.1097-1101, Jan 2018.
[4] D. Kim and J. E. Hong, "Improving software artifacts reusability based on context-aware reuse technique," Journal of Theoretical and Applied Information Technology, Vol. 96, No. 2, pp.523-533, Jan 2018.
[5] D. Kim, et al., "A dynamic control technique to enhance the

flexibility of software artifact reuse in large-scale repository," The Journal of Supercomputing, pp.1-31, June 2018.
[6] S. Hwang, et al., "System for the Research Map to Promote Convergence Research," Advances in Computer Science and Ubiquitous Computing, pp.1168-1173, Jan 2018.
[7] K. Pohl, et al., "Software Product Lines," The Essence of Software Engineering, pp.185-201, June 2018.
[8] R. Mitschke and M. Eichberg, "Supporting the Evolution of Software Product Lines," ECMDA Traceability Workshop, pp.87-95, June 2008.
[9] D. Ko, et al., "Resource versioning scheme in evolutionary software product line," International Journal of Software Engineering and its Applications, Vol. 8, No. 2, pp.113-126, Jan 2014.
[10] W. Maalej, et al., "Using contexts similarity to predict relationships between tasks," The Journal of Systems and Software, Vol. 128, pp.267-284, June 2017.
[11] Wikipedia contributors. (2018, October 10). IEC 61508. In Wikipedia, The Free Encyclopedia. Retrieved 17:45, December 16, 2018, from https://en.wikipedia.org/w/index.php?title=IEC_61508&oldid=863370057
[12] Wikipedia contributors. (2018, Dec. 2). Jaccard index. In Wikipedia, The Free Encyclopedia. Retrieved 06:03, December 13, 2018, from https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=871576429
[13] Wikipedia contributors. (2018, Nov. 14). Lexicographical order. In Wikipedia, The Free Encyclopedia. Retrieved 06:11, December 13, 2018, from https://en.wikipedia.org/w/index.php?title=Lexicographical_order&oldid=868871541

기계학습 기반 자율 제어 시스템의 참조 아키텍처

송명호○, 김수동*
숭실대학교 컴퓨터학과
{songmho, sdkim777}@gmail.com

A Reference Architecture for Autonomous Systems with Machine Learning

Myeong Ho Song○, Soo Dong Kim*
Department of Computer Science and Engineering, Soongsil University

요 약

자율 컴퓨팅은 자동화 시스템에 자율 환경 인지, 자율 판단, 자율 계획, 자율 제어 기능을 제공하는 미래기술이다. 그러나 본 기술이 특정 기능이나 시스템에만 적용되고, 높은 기술적 난도와 낮은 개발 생산성의 문제를 가지고 있다. 본 논문에서는 자율화 시스템 개발에 공통으로 적용될 수 있는 아키텍처 모형, 핵심 컴포넌트, 주요 알고리즘 등을 포함하는 참조 아키텍처를 제시한다.

1. 서 론

자율(Autonomous) 컴퓨팅은 자동화(Automatic) 시스템에 자율 환경 인지, 자율 판단, 자율 계획, 자율 제어 기능을 제공하는 미래기술이다. 현재 스마트 팩토리 [1], 자율주행 자동차 [2][3] 등 여러 분야에서 연구가 진행되고 있다. 하지만 대다수 연구는 특정 기능이나 시스템에만 적용되고, 높은 기술적 난도와 낮은 개발 생산성의 문제를 가지고 있다.

본 연구는 IBM의 자율화 지원 등급제(Evolutionary Levels) [4] 중 Level 5 자율 제어 시스템의 참조 아키텍처를 제안한다. 본 참조 아키텍처는 목표 시스템의 구조적 근간으로 활용되며, 컴포넌트, 디바이스 연동 등에 대한 설계 재사용으로 개발 효율성을 크게 높이며, 자율화 관점의 높은 품질을 보장한다.

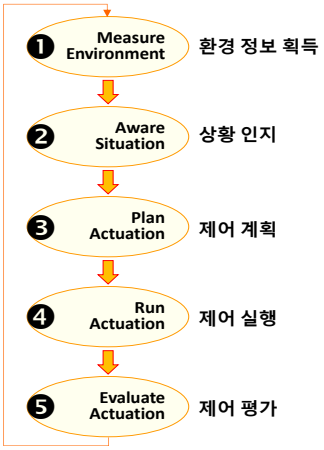


그림 1. Control Flow of Autonomous System

2. 자율화 시스템의 제어 프로세스

본 장에서는 참조 아키텍처 설계를 위해, 그 근간이 되는 자율화 프로세스를 그림 1과 같이 정의한다. 이 프로세스는 센서와 디바이스 등으로 환경 정보를 획득하고, Noise 정보를 제거한 후 (Task 1), 상황 프로파일(Situation Profile)을 통해 상황을 인지한다 (Task 2). 그 후, 제어 프로파일(Actuation Profile)을 통해 인지된 상황에 적합한 환경 제어를 결정(Task 3) 및 실행(Task 4)하여 해당 제어의 결과를 평가(Task 5)한다.

3. 참조 아키텍처 설계

3.1. 아키텍처 구조적 모델

그림 2은 참조 아키텍처의 계층과 구성요소를 나타낸다.

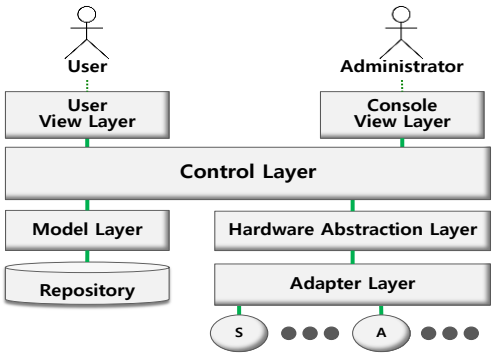


그림 2. Layers of the Reference Architecture

참조 아키텍처의 구조 모델을 설계하기 위하여, 다양한 아키텍처 스타일(Style)을 분석하였고, 그 결과 Layered 아키텍처 스타일과 하드웨어 추상 계층(HAL)을 적용한 모델이

* 교신 저자 (Corresponding Author)

도출되었다. 제시된 참조 아키텍처의 구조적 모델은 대부분의 자율 제어 애플리케이션 시스템 개발에 범용적으로 적용되고 소프트웨어 아키텍처 설계의 원칙을 적용하여 설계되었다.

3.2. 아키텍처 기능 (Functional) 뷰 모델

그림 3은 참조 아키텍처의 Control Layer에 포함되는 자율 제어와 기계학습 모델 생성을 위한 컴포넌트들을 보여준다.

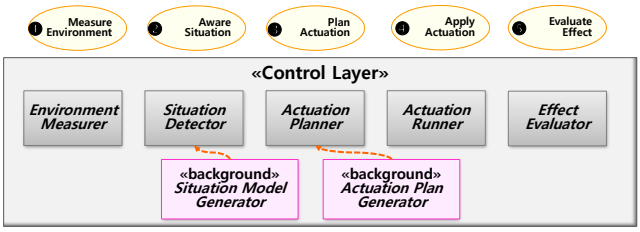


그림 3. Functional Components of the Architecture

그림 3의 상단부에 있는 5개 컴포넌트들은 제안한 제어 프로세스의 기능을 수행한다. 그림 3의 하단부의 두 컴포넌트들은 머신 러닝을 통한 상황 프로파일과 제어 프로파일 모델을 생성한다. 이 두 컴포넌트는 별도의 스레드(Thread)를 통해 위의 컴포넌트들과 병렬로 실행되고, 본 아키텍처에서 완전 자율 (Full Autonomic)을 지원하게 된다.

3.3. 아키텍처 행위 (Behavior) 뷰 모델

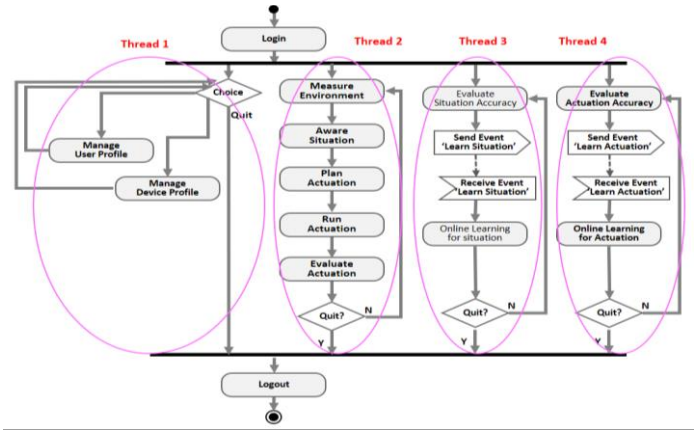


그림 4. Control Flow of Autonomous Systems

참조 아키텍처 기반 애플리케이션의 행위 뷰를 Activity Diagram으로 표현하면 그림 4과 같다. 자율 제어 시스템의 Control Flow는 네 개의 스레드(Thread)가 병렬로 실행된다.

Thread 1은 사용자가 직접 실행하는 디바이스 연결, 환경 설정 등이 해당된다. Thread 2는 앞에 제시된 자율 제어 프로세스의 Task들이 순차적으로 반복하여 수행하는 Closed Loop이다. 이 Task들은 Pile-and-Filter 방식을 통해 컴포넌트들 간 병렬 처리가 가능하다. Thread 3과 Thread 4는 환경 모델과 제어 모델의 정확도가 낮을 경우, 온라인(Online) 기계 학습을 통해 이 모델들이 자율적으로 정제 및 최적화를 한다. 따라서, 자율 제어 시스템의 서비스 품질이 향상된다.

3.4. 높은 자율성(Atomicity) 품질을 위한 설계

참조 아키텍처는 항상 높은 정확도를 유지하기 위해 아래 경우 중 한 경우 이상 만족할 경우, 상황 모델과 제어 모델에 대해 실시간 훈련(Online Learning)을 한다.

- Database에 일정량의 data가 저장되었을 경우
- 저장된 데이터에 대해 모델 정확도가 낮을 경우

자율성 품질을 향상시키는 기능을 Activity Diagram으로 표현하면 그림 5과 같다. 이를 통해 해당 아키텍처는 높은 정확도를 유지할 수 있으며 자율성 품질 또한 높게 유지한다.

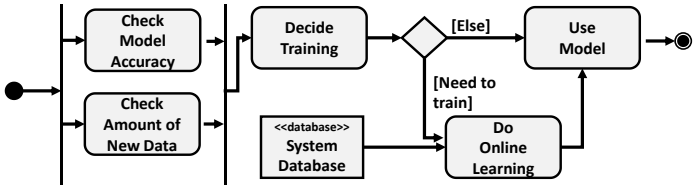


그림 5. 자율성 품질 유지를 위한 Activity Diagram

4. 실험

현재 사람의 살아있는 정도를 판단하는 People Aliveness 탐지 시스템과 차량과 행인의 밀집도를 통해 신호등 주기를 조절하는 Traffic Control 시스템에 본 자율 제어 아키텍처 적용 유무에 대한 실험을 진행하였다. 시스템이 사용하는 데이터셋이 커질수록 두 경우의 정확도는 증가하였지만, 데이터셋이 증가할수록 아키텍처 적용 시와 미 적용 시의 차이가 더욱 크게 증가하였다.

5. 결론

본 논문은 자율 제어 애플리케이션에 범용으로 적용될 수 있는 참조 아키텍처를 제안하고, 자율 제어 프로세스와 이를 지원하기 위한 구조, 기능, 행위 컴포넌트, 자율성 품질 보장 설계를 제시하였다. 이 아키텍처를 통해 기술적 난도와 설계 비용을 줄이고 높은 자율 제어 시스템 개발이 가능하다.

6. 참고문헌

[1] JunPing Wang, YunChuan Sun, and et al., “Large-Scale Online Multitask Learning and Decision Making for Flexible Manufacturing,” *International Journal of IEEE Transactions on Industrial Informatics,* Vol. 12, No. 6, pp. 2139-2147, April 2016

[2] Yi Zhang, Ping Sun, and et al., “Human-like Autonomous Vehicle Speed Control by Deep Reinforcement Learning with Double Q-Learning,” *In proceedings of IEEE Intelligent Vehicles Symposium (IV 2018),* Changshu, China, *pp. 1251-1256,* June 2018

[3] Jie Tang, Liu Shaoshan, and et al., “Teaching Autonomous Driving Using a Modular and Integrated Approach,” *In proceedings of IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC 2018)*, Tokyo, Japan, pp. 361-366, July 2018

[4] R. Murch, “Autonomic Computing”, Prentice-Hall, pages pp. 0–20:25–40, October 2004

OpenDS를 이용한 자율주행소프트웨어 시뮬레이션

윤유상○, 이채은, Nazakat Ali, 홍장의
충북대학교 소프트웨어학과
yusang@selab.cbnu.ac.kr, Jehong@chungbuk.ac.kr

Simulation of Autonomous Vehicle Software Using OpenDS

Yusang Yun, Chae Eun Lee, Nazakat Ali, Jang-Eui Hong
Dept. of Computer Science, Chungbuk National University

요 약

자율주행자동차는 다양한 센서 및 영상인식 장치, 구동기 등을 이용하여 인간의 간섭없이 주행하는 자동차이다. 따라서 자율주행 소프트웨어는 실제환경에서 사고의 발생없이 정확하게 동작해야 하기 때문에 철저한 테스트가 요구된다. 그러나 실제 세계에서 자율주행소프트웨어를 테스트하는 것은 물리적인 손상을 초래할 수 있기 때문에 가상의 시뮬레이션을 통해 소프트웨어의 동작에 대한 확인하는 것이 매우 중요하다. 본 논문은 자율주행 소프트웨어의 동작이 의도한 대로 이루어지고 있는가를 확인하기 위한 목적으로 OpenDS를 이용한 자율주행소프트웨어 시뮬레이터를 개발하였다. 특히 시뮬레이션이 웹에서 가능하도록 구현함으로써, 시뮬레이터를 가지지 않은 개발자도 쉽게 활용할 수 있는 환경을 제공하였다.

1. 서 론

자율주행차에 대한 연구 개발이 활발해지고 있는 상황에서 구글, 네이버 등과 같dl 자율 주행을 위한 인프라를 구축하는 기관과, GM, 테슬라 등과 같은 자동차 전문회사들이 많은 자율주행차를 개발하고 시험운행을 추진하고 있다.

그러나 이러한 많은 투자와 연구 개발에도 불구하고 자율주행차가 사고를 유발하는 기사를 종종 보게 된다. 이는 자율주행을 제어하는 소프트웨어가 아직은 많은 개선점을 지니고 있다는 것을 의미한다. 자율주행 소프트웨어를 개발함에 있어서 극복해야 하는 중요한 기술적 이슈[1,2]들은 크게 다음과 같이 정리할 수 있다.

● 실시간성 : 자율주행은 실제 환경에서 수행되는 행위이기 때문에 발생하는 모든 주행 이벤트에 대하여 실시간 처리가 이루어져야 한다.

● 불확실성 : 실 주행환경에서 발생하는 모든 상황을 소프트웨어 개발 시점에서 고려하기 어렵기 때문에 이에 대한 적절한 처리가 요구된다.

● 의사결정성 : 소프트웨어 개발 시 고려되지 못했던 상황이 발생했을 때, 이를 해결하기 위한 최적의 방안이 결정되어야 한다.

● 가시성 : 자율주행 과정에서 발생하는 모든 정보들이 모니터링되고, 이들은 종합하여 인간이 인지 가능한 형태의 가시적인 정보로 제공되어야 한다.

● 강제종료 : 어떠한 경우 또는 상황에서도 자율주행 소프트웨어의 강제적인 종료가 가능하도록 하는 HIL (Human in the Loop) 메커니즘이 지원되어야 한다.

이와 같은 자율주행 소프트웨어 개발의 핵심적인 고려요소들은 인간의 생명이나 재산상의 손실에 직접적인 영향을 줄 수 있는 요소들이기 때문에 안전성 측면에서 매우 중요하게 고려되어야 하는 요소이다[3]. 그러나 자율주행 소프트웨어의 개발은 테스트 과정이 복잡하고 다양해서 실제로 필드 테스트가 용이하지 않다[4]. 많은 시간과 비용이 투자되고 있는 필드 테스트에서 많은 기존의 자율주행차가 사고를 유발하고 있다.

따라서 본 연구에서는 자율주행 과정에서 발생할 수 있는 여러 가지 상황을 시나리오로 정의하고 테스트할 수 있는 시뮬레이션 시스템을 개발한다. 특히 자율주행과 관련된 다양한 객체들을 라이브러리로 제공하는 OpenDS[5]를 이용하여 개발한다.

OpenDS 기반의 자율주행 소프트웨어의 시뮬레이터는 개발 플랫폼을 활용함으로써, 구현이 간단하고, 다양한 시나리오 테스트가 가능하다는 장점을 제공한다. 본 논문의 2장에서는 자율주행 소프트웨어, 자율주행의 수준 등에 대한 기반 정보를 설명하고, 3장에서는 기존의 시뮬레이션 시스템을 분석하고, 4장에서는 이를 바탕으로 시뮬레이션 시스템의 구성, 5장에서는 자율주행 시나리오의 시뮬레이션 과정을 설명한다. 6장에서는 결론 및 향후 연구 내용에 대하여 기술한다.

2. 배경 지식

2.1 자율주행 수준

현재 자율주행차는 NHTSA(미국 도로 교통 안전청)에 따라 표 1과 같이 5개의 레벨로 구분된다[1]. 현재 자율주행차는 Level 2 수준의 지능형 운전자 보조시스템에 해당한다. Level 2 수준은 여러 개의 기능을 자동화하여 복합적인 운행이 가능한 단계이며 일부 상용화가 진행중이다. 또한 2020년까지 자동차가 직접 주행을 하며 특정 상황에서 수동 운전으로 전환되는 Level 3 수준을 목표로 하고 있다.

표 1. 자율주행차의 자동화 단계 구분

자동화 단계	정의	내용	운전 주체
Level 0	No Aytimation (비자동)	· 운전자가 항상 수동으로 조작 · 현재 생산되는 대다수의 자동차가 이 단계에 해당	운전자
Level 1	Function Specific Automation (기능제한 자동화)	· 자동 브레이크와 같이 운전자를 돕는 특정한 자동제어기술이 적용	운전자
Level 2	Combined Function Automation (복합기능 자동화)	· 두 가지 이상의 자동제어기술 적용 · 차선유지시스템이 결합된 크루즈 기능이 이에 해당 · 일부 상용화 진행 중	운전자
Level 3	Limited Self-Diving Automation (제한된 자동화)	· 고속도로와 같은 일정 조건 하에서 운전자 조작 없이 스스로 주행 가능 · 돌발 상황에서 운전자의 개입이 필요	시스템/ 운전자
Level 4	Full Self-Driving Automation (완전 자동화)	· 운전자가 목적지와 주행경로만 입력하면 모든 기능을 스스로 제어해서 수행 · 운전자의 개입 불필요	시스템

2.2 자율주행 소프트웨어

자율주행 소프트웨어는 표 2와 같은 기술을 바탕으로 하드웨어와 연동하여 자율주행차를 제어하는 역할을 담당한다[1,6].

표 2. 자율주행 소프트웨어 관련 기술

SW 기술	설명
센서 정보 기술	센서 정보를 수집 및 분석
영상 정보 기술	영상 정보를 수집 및 분석
상황 인식 기술	센서 및 영상 정보를 바탕으로 주행에 필요한 상황 인식
방향 제어 기술	인식한 상황을 바탕으로 주행 속도 및 방향 제어
경로 계획 기술	자율 주행을 위한 목적지까지의 경로 계획 및 지도 작성
인공 지능 기술	날씨나 차량 주변 환경 등의 변수를 학습

센서 정보 기술은 부착된 센서에서 수집된 데이터를 분석하여 주변 환경에 대한 정보를 수집할 수 있는 기술을 말한다. 영상 정보 기술은 카메라를 통해 입력되는 영상을 분석하여 차선이나 도로의 궤적 정보, 신호등 정보, 주변 환경 정보를 수집할 수 있는 기술을 의미한다.

상황 인식 기술은 센서와 카메라를 바탕으로 수집된 주변 환경 데이터를 통해 상황을 인식하여 알맞은 주행을 위한 의사결정을 하는 기술이다.

방향 제어 기술은 인식한 상황을 바탕으로 차량의 주행 속도를 감속·가속하거나, 방향을 제어하는 기술을 말한다.

경로 계획 기술은 처음에 지정된 목적지까지의 최적 경로를 선택하고 이를 지도로 표현할 수 있는 기술을 의미한다.

인공 지능 기술은 날씨나 사고와 같은 예상치 못한 환경에 대해 추가적으로 학습하여 주행에 반영할 수 있는 기술을 말한다.

3. 자율주행소프트웨어 시뮬레이션 시스템의 특징

시뮬레이션 시스템은 자율주행소프트웨어의 개발 및 테스트에 사용되는 도구로서, 표 3은 현재 상용화된 자율주행소프트웨어 시뮬레이션 시스템을 분석한 결과이다. 이를 바탕으로 자율주행소프트웨어 시뮬레이션 시스템의 특징을 표 3과 같이 7가지로 정리한다[4].

표 3. 기존 시뮬레이션 시스템 분석

Name	R1	R2	R3	R4	R5	R6	R7
MontiSim	√	√	√	Win/Lin/Mac/Web	√	√	√
Pelops			√			√	√
Prescan	√	√	√	Win		√	√
Mathworks		√	√	Win/Lin/Mac/Web			√
RT Maps			√	Win/Lin			√
Gazebo		√	√	Win/Lin	√	√	√
SUMO	√	√		Win/Lin			
PTV Vissim	√	√		Win			√
SimTram	√	√		Win/Lin			√
Vires VTD		√	√	Lin		√	√
Udacity			√	Win/Lin/Mac			√
OpenDaVinci		√	√	Win/Lin	√		√
CarSim		√	√	Win	√	√	√

R1. 현실 세계 반영

자율주행소프트웨어를 시뮬레이션 하기 위해서는 실제와 같은 현실감을 갖는 환경이 중요하다. 이를 위해서 현재 많은 시뮬레이션 프레임워크들은 복잡한 맵을 제공하거나 OpenStreetMap과 같은 외부 모듈을 사용함으로써 실제 주요 도시의 환경을 구축하는 기능을 제공한다.

R2. 일상 시뮬레이션 지원

자율주행소프트웨어의 개발 및 테스트를 수행하기 위해서 시뮬레이션 시스템은 다양한 환경을 지원해야 하며, 대규모 시나리오를 제공할 필요성이 있다. 그리고 다양한 환경을 지원하기 위해서는 다수의 차량 설정, 교통밀도 설정, 기상 조건 설정, 보행자 여부 설정 등을 지원해야 한다.

R3. 컨트롤러 및 자동차 모델 지원

컨트롤러를 갖춘 시뮬레이션 시스템은 자율주행소프트웨어에게 다양한 조건을 부여할 수 있다. 또한, 자율주행 소프트웨어의 전체적인 동작뿐만 아니라 특정 모듈의 동작을 확인할 수 있다.

그리고 다양한 자동차 모델의 지원은 여러 모델의 시뮬레이션을 가능하게 한다. 이는 자율주행소프트웨어를 다양한 경우에 대해 학습 및 테스팅할 수 있게 한다.

R4. 다중 플랫폼 지원

시뮬레이션의 용이함을 위해서 시뮬레이션 시스템은 다양한 환경에서 동작할 수 있어야 한다. 동작 환경에는 Windows, Linux, Mac, Web 등이 존재한다.

R5. 지속적인 통합 및 회귀 테스트 지원

자율주행소프트웨어를 모듈 단위로 개발하기 위해서 현재의 시뮬레이션 시스템 중 지속적인 통합과 회귀 테스트를 지원하는 경우가 존재한다. 이는 자율주행소프트웨어의 품질을 높여주며, 알고리즘 단위의 테스트를 용이하게 한다.

R6. 물리 엔진 지원

시뮬레이션의 현실감을 높이기 위해, 시뮬레이션 시스템은 물리 엔진을 지원해야 한다. 또한 이를 통해 주행 중 사고 발생 시, 자율주행소프트웨어의 반응을 학습 및 테스트할 수 있다.

R7. 3D Visualization 지원

3D 시뮬레이션을 제공하기 위해 대부분의 자율주행 소프트웨어 시뮬레이션 시스템은 3D Visualization을 지원한다.

본 논문에서 다루고 있는 자율주행소프트웨어 시뮬레이션 시스템은 3D 주행 시뮬레이터인 OpenDS를 활용함으로써 R1, R2, R6, R7을 기본적으로 제공하며, 컨트롤러 시스템을 통해 R3, R4의 특징을 추가적으로 제공하고자 한다.

4. OpenDS 기반 자율주행 시뮬레이션 시스템

4.1 OpenDS 플랫폼

OpenDS 은 주행 시뮬레이션 시스템으로 고가의 상용 시뮬레이션 시스템과 달리 JAVA 기반의 오픈 소스 플랫폼으로 무료로 사용할 수 있다. OpenDS 주행 시뮬레이션 시스템은 자율주행소프트웨어 시뮬레이션 시스템으로 활용할 수 있는 특징을 가지고 있으며, 이는 다음과 같이 정리한다[5,7].

4.1.1 다양한 포맷의 맵 데이터 지원

OpenDS는 다양한 맵을 지원한다. OpenDS의 오리지널 맵 데이터는 xml 형식이기 때문에, 사용자 임의대로 수정 및 삭제가 가능하다. 이외에도 외부 모듈인 OpenStreetMap의 OSM 데이터를 통하여 실제 도시 환경의 구축이 가능하며, 그림 1은 이러한 방식으로 Berlin의 Treptow 도시를 구축한 화면이다. 이를 통하여 실제와 비슷한 환경에서 자율주행 소프트웨어 테스팅이 가능하다. 또한, 경우에 따라서 사용자 임의대로 맵 데이터를 조작 및 변경하여 필요한 환경을 구축하여 학습 및 테스팅이 가능하다.

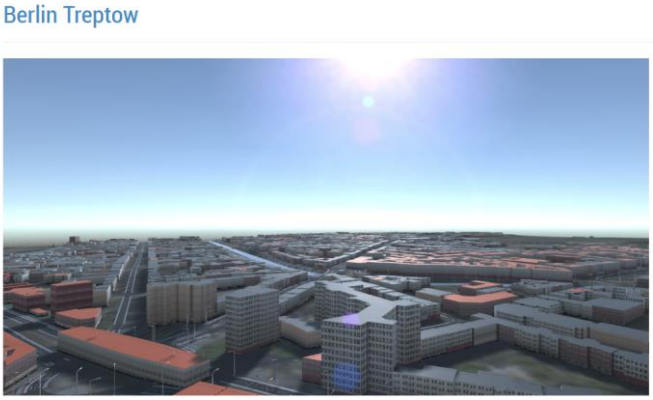


그림 1. OpenDS 내 Berlin의 Treptow 맵

4.1.2 시나리오 요소 지원

OpenDS는 일상에 가까운 환경을 제공하기 위해 교차로 및 신호등, 다수 차량의 주행, 비 또는 안개와 같은 기상 조건, 보행자 등과 같은 여러 시나리오 요소의 설정 기능을 지원한다.

이러한 구성 요소의 제공은 자율 주행에 필요한 여러 가지 요소들이 실제 주행에서 영향을 미치는 정도를 확인하고, 또한 이들을 탐지하여 주행이 가능한지를 확인하기 위해 제공하는 요소들이다. 그림 2는 안개가 끼는 기상 조건을 부여한 환경(좌)과 주행 시나리오가 입력된 차량 설정 화면(우)이다.

그림 2. OpenDS 내 안개 설정(좌)와 주행 설정(우)

4.1.3 물리 엔진 적용

OpenDS는 보행자나 차량 등과 같은 오브젝트 사이에서 발생하는 상호작용에 대해서 물리 엔진이 적용된다. 이 때문에 사고 상황에 대한 차량 및 보행자의 반응을 현실 세계와 유사한 형태로 확인할 수 있다. 그림 3은 차량 간 충돌에 대한 시나리오 테스트 화면이다.

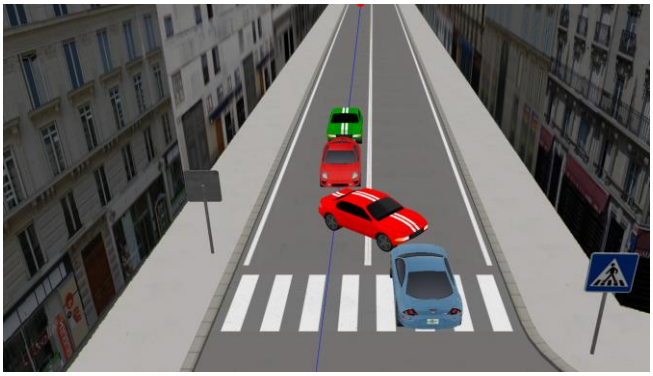


그림 3. OpenDS의 차량 충돌 시나리오 테스트 화면

4.2 시뮬레이션 시스템 아키텍처

본 시뮬레이션 시스템은 그림 4와 같이 Simulation Controller와 Vehicle Simulator로 구성된다. 또한 이 두 요소는 각각 다른 PC에서 구동되며, TCP 통신으로 연결되어 데이터를 공유한다.

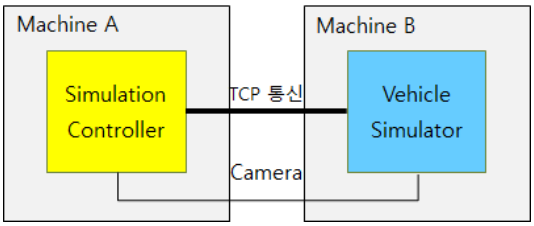


그림 4. 시뮬레이션 시스템 구조

Vehicle Simulator는 TCP 통신을 통해 전달받은 자동차의 속도 및 방향 데이터를 바탕으로 자율주행차를 시뮬레이션 하는 기능을 담당한다. 본 시스템에서는 OpenDS를 통해 해당 기능을 수행한다.

Camera는 Vehicle Simulator의 화면 데이터를 Simulation Controller에게 전송함으로써 OpenDS 내의 정보를 Simulation Controller에 전송한다. 또한 Vehicle Simulator와 Simulation Controller 간의 자동차의 방향 및 속도 데이터는 TCP 통신을 통해 공유된다.

Simulation Controller는 주변 환경을 인식하여 자동차의 방향 및 속도 제어하는 역할을 담당한다. 이러한 Simulation Controller의 자세한 구조는 그림 5처럼 Lane Control Module과 Driving Control Module로 구성된다.

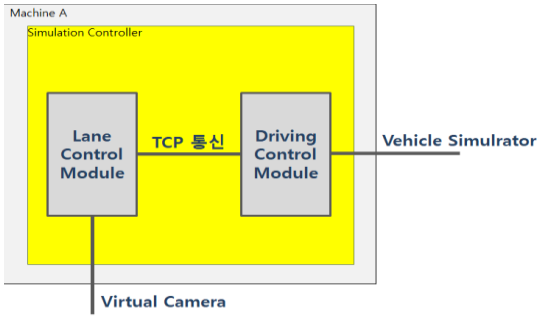


그림 5. Simulation Controller 구조

Lane Control Module은 Camera를 통해 전달받은 영상 데이터를 바탕으로 차선 정보, 신호등 정보, 보행자 정보와 같은 주변 환경 정보를 얻는 기능을 담당한다. 이러한 정보는 TCP 통신을 통해 Driving Control Module로 전달된다.

Driving Control Module은 Lane Control Module로부터 전달받은 환경 정보를 바탕으로 알맞은 주행 속도와 방향을 계산하여 Vehicle Simulator에 전달한다.

4.3 시뮬레이션 시스템 동작구성

본 시뮬레이션 시스템의 전체적인 동작 알고리즘과 Sequence Diagram은 각각 그림 6과 같다[5].

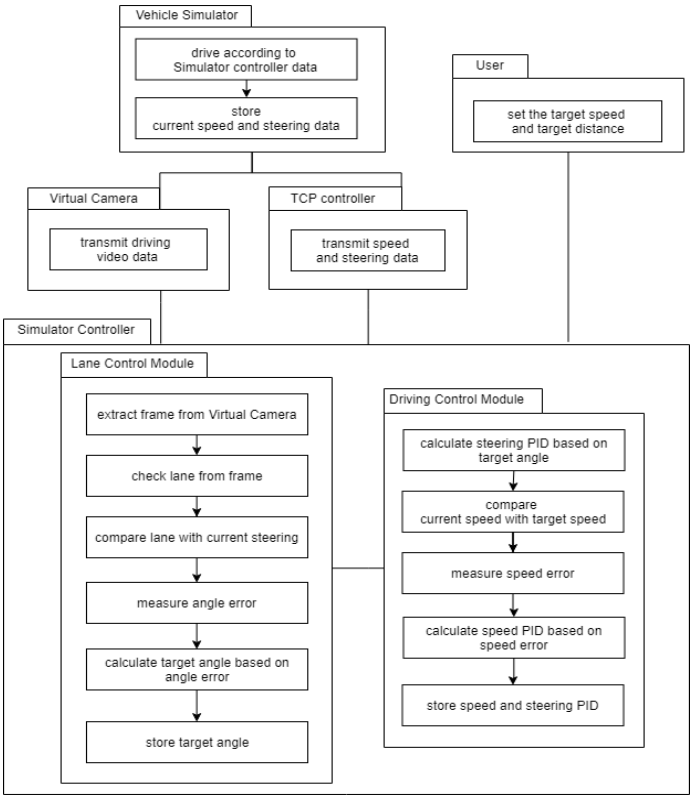


그림 6. 시뮬레이션 시스템 전체 동작 과정

User는 먼저 자동차의 속도와 주행 거리 등과 같은 주행에 필요한 초기값을 입력한다. 이렇게 입력 받은 데이터는 Simulator Controller를 통해 Vehicle Simulator로 전송된다. Vehicle Simulator는 전송 받은 속도와 방향 데이터를 바탕으로 자동차 주행 시뮬레이션을 시작한다. Vehicle Simulator에서 제공하는 주행 영상과 현재 주행 정보는 각각 Camera와 TCP Controller를 통해 Simulator Controller로 전송된다.

Simulator Controller는 이렇게 영상과 주행 정보를 바탕으로 다음과 같은 동작을 수행한다. Simulator Controller 내부의 Lane Control Module은 전송 받은 영상을 바탕으로 차선을 검출한다. 이후 검출된 차선 정보

와 차량의 주행 방향을 비교하여 target angle를 계산한 뒤, 이를 Driving Control Module에 전송한다. Driving Control Module에서는 target angle을 바탕으로 steering PID(Proportional Differential Integral)를 계산하고, target speed와 주행 속도 정보를 바탕으로 speed PID를 계산한다. 이렇게 계산된 두 PID 정보는 TCP Controller를 통해 Vehicle Simulator에 전송되고, Vehicle Simulator는 이 정보를 바탕으로 차량 주행을 진행한다.

4.4 거리 제어 알고리즘

자율주행에서의 차간 거리의 유지는 충돌 예방을 위하여 중요한 부분이다. 속도의 제어를 통해 차간의 거리에 따라 제어하게 되는데, 그 개념은 그림 7과 같다.

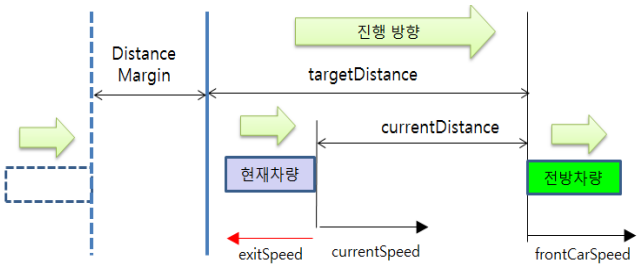


그림 7. 자율주행에서의 거리제어 개념

즉, 현재의 속도(currentSpeed)와 거리(currentDistance)를 고려하여 속도 제어 정보를 산출한다. 또한 속도의 감속은 후방의 차량에도 영향을 주기 때문에 앞 차량과의 정상 유지 거리(targetDistance)와 후방 차량과의 여유 거리(distanceMargin)도 고려되어야 한다. 전체적인 거리 제어 알고리즘은 그림 8과 같다.

```
float MainWindow::distanceControl()
{
    float currentDistance = canCtrl->getRxRadarRange();
    float currentSpeed = canCtrl->getRxCurrentSpeed();

    float timeToExit = 1 * currentDistance/currentSpeed;
    float timeToEnter = 0.5;
    float distanceMargin = currentSpeed * timeToEnter;
    float err = targetDistance + distanceMargin - currentDistance;
    float exitSpeed = 3.6 * (err / timeToExit);

    if(currentDistance >0 && err > 0)
    {
        float newTargetSpeed = currentSpeed - exitSpeed;
        if(newTargetSpeed < 0) newTargetSpeed = 0;
        if(newTargetSpeed < targetSpeed) return newTargetSpeed;
    }
    return targetSpeed;
}
```

그림 8. 자율주행에서의 거리제어 알고리즘

그림 8에서 보여주는 것과 같이 변수, targetDistance, distanceMargin, 그리고 currentDistance를 이용하여 속도 오류(err) 값을 산정하고, 이 값이 0보다 크면 속도를 감속(exitSpeed)하여 목표 속도(newTargetSpeed)를 산정하게 된다.

5. 시나리오 시뮬레이션

5.1 시뮬레이션 구성

본 연구에서는 시뮬레이션 시스템을 바탕으로 시나리오 시뮬레이션 결과를 제시한다. 시나리오 시뮬레이션은 특정 상황의 흐름을 기반으로 수행한다. 시나리오 수행은 자율주행이 정상적으로 진행되는지를 확인하는 과정이다. 그림 9는 개발된 시뮬레이터의 화면을 나타낸 것이다.

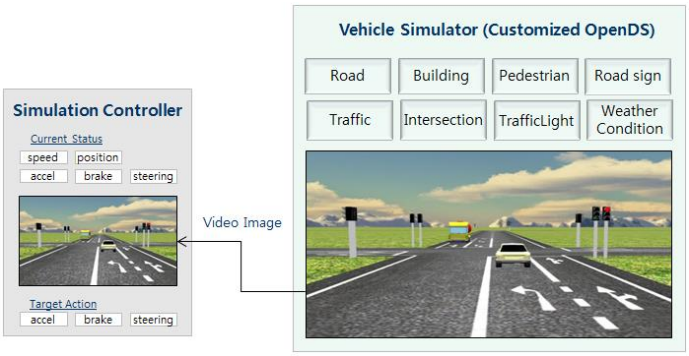


그림 9. OpenDS 기반 자율주행 시뮬레이터

OpenDS를 이용하여 구축한 시뮬레이션 시스템은 그림 9와 같이 Simulation Controller는 현재의 주행 상태에 대한 정보를 제공하는 부분과 이를 기반으로 사용자가 가속이나 감속, 방향 전환 등을 직접 제어할 수 있는 부분으로 구성된다.

또한 Vehicle Simulator에서는 주행을 위한 도로 및 환경 요소들을 선택하여 주행 맵을 구성할 수 있도록 지원하는 부분과 구축된 주행 맵에서 자율 주행이 진행되면, 이때 화면을 Controller에게 전달하여 주행 상태를 확인하게 된다.

5.2 시뮬레이션 시나리오

자율주행 차에 대한 시나리오 시뮬레이션은 정상 시나리오와 비정상 시나리오를 모두 검증해야 한다. 정상적으로 동작하는 부분에 대한 검증은 일반적으로 예상대로 진행되지만, 자율 주행 소프트웨어 개발과정에서 예상하지 못했던 돌발 상황도 대처할 수 있는지를 확인하기 위하여 비 정상적인 주행 시나리오를 선정하여 시뮬레이션을 수행한다[8].

시뮬레이션을 위해 고려하는 정상적인 주행 시나리오와 비 정상적인 주행 시나리오는 다음과 같다.

<시나리오 #1 : 속도 제어>
[정상 주행] 제한속도 60Km 구간에서 60Km 이하의 속도를 주행한다.
[비정상 주행] 제한속도 60km 구간에서 가속을 시도하여 70Km로 주행한다.

<시나리오 #2 : 신호 인식>
[정상 주행] 교차로 신호등이 붉은색인 경우 정차하도록 한다.
[비정상 주행] 교차로 신호등이 녹색인 경우에도 정차하도록 시도한다.

제시하는 첫 번째 시뮬레이션 시나리오에서 정상적인 경우에는 자율 주행 소프트웨어의 동작에 의거하여 수행됨을 관찰하는 것이고, 비정상 주행의 경우에 있어서는 사용자의 간섭에 의해 Simulation Controller에서 목표 속도를 기준보다 높이는 경우이다. 이러한 경우에도 Vehicle Simulator에서 탐지한 OpenDS 주행 환경에 근거하여 주행하는지를 확인하고자 하는 것이다. 즉, 일시적인 가속이 발생했지만, 곧 정상적인 규정 속도로 주행 속도가 변경되는지를 확인하는 것이다.

두 번째 시나리오의 비정상적인 경우는 녹색 신호등이 점등된 교차로에 진입시 사용자가 Simulation Controller에서 브레이크를 작동시키는 것을 시도하는 경우로, 이러한 상황에서도 브레이크 해제시 곧 신호등을 탐지하여 주행하는지를 확인하는 것이다.

5.3 시뮬레이션 결과 및 고찰

자율주행 차에 대한 시나리오 시뮬레이션은 정상 시나리오와 비정상 시나리오로 구분하여 수행한다. 이는 비정상적인 상황에서도 자율주행 차가 올바른 동작을 수행할 수 있는 가, 또는 상식적이고 근거 있는 동작을 수행하는 가를 판단하기 위함이다.

자율주행 차가 스스로 대처하지 못하는 상황을 대비하기 위하여 사람의 간섭이 유효하도록 소프트웨어가 개발되어야 하며, 주행 상황이 안정적인 상태로 돌아 왔을 때, 정상적인 주행으로 복귀해야 한다.

본 연구에서의 시나리오는 아주 단순한 경우이지만, 매우 복잡하고 쉽게 판단하기 어려운 주행 상황이 존재할 수 있음은 익히 알려져 있다. 운전자를 보호할 것인가, 보행자를 보호할 것인가에 대한 우선순위 결정부터 주행 기준을 위반하는 다른 차량에 대한 대처까지 매우 다양하다. 이러한 상황에서도 자율 주행 소프트웨어가 정상적으로 동작할 수 있는 지는 시뮬레이션에서 다루어야 하는 중요한 부분이다[9].

6. 결론 및 향후 연구내용

본 논문에서는 주행 시뮬레이션 시스템인 OpenDS를 이용하는 자율주행소프트웨어 시뮬레이션 시스템을 제시한다. 또한 이를 바탕으로 정상적인 주행 시나리오와 비정상적인 주행 시나리오를 통해 자율주행 소프트웨어를 확인하는 시나리오 시뮬레이션에 대하여 설명하였다.

자율주행 소프트웨어는 인간의 간섭 없이 자율적으로 주변 환경을 인식하여 자체적으로 의사결정을 하고, 이 결과를 바탕으로 주행하는 소프트웨어를 말한다. 즉, 자율주행 소프트웨어는 주행 중 발생하는 일반적인 상황뿐만 아니라 특수한 상황을 포함한 모든 상황에 대해 대처할 능력을 가지고 있어야 한다. 이를 위해서 본 논문에서는 다양한 주행 환경을 구축할 수 있도록 OpenDS를 이용하여 주행 환경을 구성하였다.

본 연구에서 제시한 시나리오 시뮬레이션을 통해 자율주행차의 사고를 예방하고 이에 따른 인명 피해와 금전적 피해를 줄일 수 있을 것으로 기대한다.

향후에는 시뮬레이션 프로그램의 기능을 확장하고 이에 따라 보다 넓은 영역의 시나리오 추가를 통해 보다 다양한 상황을 시뮬레이션할 수 있도록 할 예정이다.

Acknowledgement

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업(NRF-2017M3C4A7066479)의 지원을 받아 수행한 연구임.

참고문헌

[1] 장승주, “자율 주행 자동차 관련 SW기술 동향”, 한국통신학회지, 33(4), pp.27-33, 2016
[2] Andrew Best, et. al., “AutonoVi-Sim: Autonomous Vehicle Simulation Platform with Weather, Sensing, and Traffic Control”, NIPS 2017 Workshop MLITS, pp.1-9, 2017
[3] J. Bischo, M. Macieje, “Simulation of city-wide replacement of private cars with autonomous taxis in Berlin,” Procedia Computer Science (83) pp. 237 - 244, 2016
[4] Filippo Grazioli, et. al., “Simulation Framework for Executing Component and Connector Models of Self-Driving Vehicles”, MODELS 2017, pp.109-115
[5] OpenDS, “OPENDS 5.0 - OPEN SOURCE DRIVING SIMULATION”, https://opends.dfki.de/, 2018.12.10
[6] J. Pereira, R. Rossetti, “An Integrated Architecture for Autonomous Vehicles Simulation,” ACM SAC’12 pp.286-292, 2012
[7] “Automated Driving Education”, ESG, 2018년 12월 15일 접속, http://www.esg-solution.com/product-autodrive/ automated-driving-education/
[8] Pascal Marcel Minnerup, “An Efficient Method for Testing Autonomous Driving Software against Nondeterministic Influences,” Ph.D Dissertation, Technical University at Munchen, Sep., 2017
[9] Rohit Ramanna, “Simulation: A Must for Autonomous Driving,” Talk-S8859, NVIDIA GTC 2018, San Jose, March 2018

시나리오 테이블을 이용한 STPA에서의 사고 원인 식별

양현수 ○, 권기현

경기대학교 컴퓨터과학과

{sean1538, khkwon}@kgu.ac.kr

Identifying causes of an accident in STPA using the scenario table

Hyunsoo Yang ○, Gihwon Kwon

Department of Computer Science, Kyonggi University

요 약

STPA(System Theoretic Process Analysis)는 안전 필수 시스템에서의 위험원 분석 기법 중 하나이다. 안전 필수 시스템에서 소프트웨어의 비중이 증가하며 등장하였으며, 사고 발생 원인 중 구성 요소 간의 상호작용을 중점으로 두어 위험원을 분석한다. STPA는 모든 사고를 제어의 관점에서 분석하고, 안전하지 않은 제어 명령이 제공되는 원인인 사고 시나리오를 식별한다. 이번 연구에서는 아직 명확한 방법론이 제시되지 않은 사고 시나리오 식별 단계에 대하여 이전 연구들에서 제시되었던 방법들의 공통점을 통합하고, 취약점을 보완하는 방법에 대한 연구를 진행한다. 이를 위하여, 사고 시나리오 식별 시 누락이 최소화되도록 보조해주는 사고 시나리오 테이블을 이용한 사고 원인 식별을 제안한다. 마지막으로 제안하는 사고 시나리오 테이블을 철도 디오라마 설계 단계에 적용하여 확인한다.

1. 서 론

최근 안전 필수 시스템의 복잡도가 증가함에 따라 소프트웨어가 차지하는 비중이 점차 증가하고 있다. 안전 필수 시스템에서 소프트웨어는 시스템을 제어하기 위한 제어 명령(Control action)을 생성하고, 제어 명령으로 발생한 피드백을 다시 입력 받는 상호 작용을 반복한다. STPA(System Theoretic Process Analysis)는 이러한 소프트웨어 중심 시스템의 위험원 분석 기법 중 하나이다. STPA는 구성 요소 간의 상호작용에 의해 사고가 발생한다는 관점에서 위험원을 분석한다. STPA의 핵심은 안전하지 않은 제어 명령을 통해 발생하는 사고의 원인을 분석하는 것이다. 그 과정에서 사고 시나리오를 식별하여 원인을 분석하고, 결과적으로 안전 요구사항을 도출해 낸다.

본 논문에서는 위험원 분석 결과의 누락을 최소화하기 위해, 기존 연구들에서 제시되었던 사고 시나리오 식별 방법들의 공통점을 통합하고, 취약점을 보완해주는 테이블 작성을 제안한다. 마지막으로, 운영 가능한 철도 디오라마 시스템에 적용하여 제안하는 연구의 유효성을 확인한다. 이는 실제 시스템을 축소하여 만든 것으로, 실제 안전 필수 시스템에서도 적용이 가능하다.

"본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학 ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2015-0-00445)

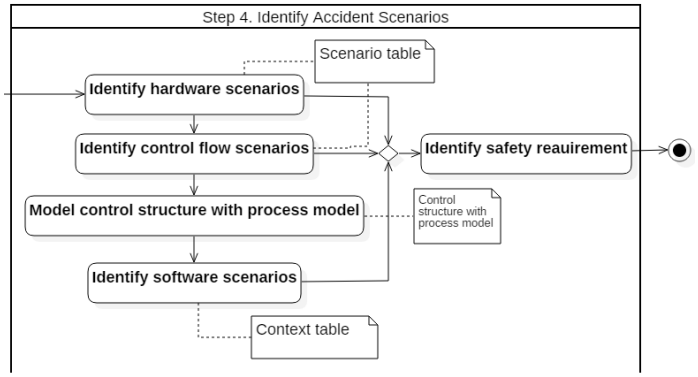
2. STPA 기법

STPA는 시스템에서 사고의 발생을 특정 구성 요소의 고장이 아닌, 복잡한 프로세스 및 구성 요소 간의 상호작용 관점으로 분류 하는 STAMP(Systems-Theoretic Accident Model and Processes) 모델을 기반으로 한다. STAMP 모델의 관점을 기반으로 안전하지 않은 제어 명령을 식별하는 것이 STPA 기법이다. 이를 위하여 제어 명령이 부적절하게 제어되는 경우를 분석한다. 이후 부적절한 제어 명령들이 발생하는 원인들을 분석한 사고 시나리오를 식별한다. 마지막으로 식별된 사고 시나리오들을 발생하지 않도록 안전 요구사항들을 도출해내는 것이 STPA의 최종 목표이다. STPA의 상세한 진행 단계는 다음과 같다[1].

Step 1: 분석을 위한 목적 정의
  Step 1.1: 분석 시스템 범위 설정
  Step 1.2: 사고 식별
  Step 1.3: 위험원 식별
  Step 1.4: 시스템 수준 안전 제약 사항 식별
Step 2: 제어 구조도 모델링
  Step 2.1: 제어 구조도 작성
  Step 2.2: 제어 명령 식별
Step 3: 안전하지 않은 제어 명령 식별
Step 4: 사고 시나리오 식별

3. STPA 사고 시나리오 식별 세부 절차

이번 절에서는 사고 시나리오 식별 누락을 최소화 하기 위한 세부 절차를 그림 1과 같이 제안한다. 이 때 기존 사고 시나리오 식별을 위한 두가지 관점[1]은 유지한 채로 시나리오 테이블에 적용할 수 있도록 한다. 첫번째 단계로 구성 요소가 물리적인 고장이 원인이 되는 하드웨어 사고 시나리오를 식별한다. 다음 단계로는 제어 명령의 전달 과정에서 발생할 수 있는 제어 흐름 사고 시나리오를 식별 한다. 마지막으로 프로세스 모델(Process model)을 기반으로 한 상황 분석 테이블(Context table)을 분석하여, 잘못된 제어 명령을 생성 할 수 있는 소프트웨어 사고 시나리오를 식별한다[1,2,3,4].



<그림 1> STPA 사고 시나리오 식별 세부절차 액티비티 다이어그램

4. 적용 결과

<표 1> 신호 제어기1의 하드웨어, 데이터 흐름 사고 시나리오 테이블 (제어기로부터 제어 명령이 제공되지 않거나, 부적절하게 제공된 경우)

	Cause		Scenario List
	Main controller	Signal controller1	
H/W	Failure	Failure	HS 1
	normal	Failure	HS 2
	Failure	normal	HS 3 HS 4
	normal	normal	HS 5 HS 6
Control Flow	Transmit	Not received	CS 1
	Not transmit	Not received	CS 2
	Not transmit	Received	CS 3
	Transmit	Received	CS 4
S/W	Process model (Control algorithm)	-	-

<표 2> 열차 위치 파악 프로세스 모델 상황 분석표

Control Action	1st Train location	2nd Train location	S/W State	Scenario list
Transmit 1st, 2nd, 3rd train location	0 ~ 12	0 ~ 12	safety	-
	0 ~ 12	unknown	hazardous	SS 1
	unknown	0 ~ 12	hazardous	
	unknown	unknown	hazardous	

적용한 시스템은 Asim의 ACC(Adaptive Cruise Control)[4]를 응용하여 열차에 적용한 MACC(Multiple Adaptive Cruse Control)이다. MACC는 3대의 열차가 설정된 폐색 구간을 유지하기 위해 서로 속도를 자동으로 조절하며 운행하는 철도 디오라마 시스템이다. 이 시스템 설계에 시나리오테이블을 적용한 결과는 표1과 표2와 같으며 식별된 시나리오의 일부는 다음과 같다.
CS 3. 선로에서 발신하지 않은 데이터(전기 신호)가 발생하여 신호 제어기1에 수신된다. [UCA1.1, UCA2.1]
SS 1. 열차 위치를 파악할 수 없어 폐색 구간 거리 계산을 할 수 없어 정확한 속도 정보를 전송할 수 없기 열차가 추돌할 수 있다. [UCA6.2]

식별된 사고 시나리오가 일어나지 않도록 도출해낸 안전 요구사항은 다음과 같다.
SR 9. 주 제어기의 열차 위치 정보에 열차 위치(1 ~ 36)이외의 정의되지 않은 값이 식별되었을 때는 모든 열차에 정지 명령을 제공하도록 한다. [CS 3, SS1]

5. 결 론

본 논문에서는 STPA의 분석 결과 누락을 최소화하기 위해 시나리오 테이블 작성을 제안하고 이를 실제로 설계 단계 적용하였다. 적용한 결과 다양하고 세부적인 사고 시나리오가 식별되었고, 연관 되는 사고 시나리오들은 서로 조합을 이루어 STPA의 최종 목표인 안전 요구사항이 도출에 기여한 것을 확인 할 수 있었다.

참고 문헌

[1] Nancy G. Levenson, John P. Thomas, "STPA HANDBOOK", 2018
[2] IPA, "First time STAMP / STPA: New safety analysis method based on system thinking (Foundation)",2016
[3] Nancy G. Levenson, John P. Thomas, "An STPA Primer v1", 2013
[4] Asim Abdulkhaleq, "Experiences with Applying STPA to Software-Intensive Systems in the Automotive Domain", 2013 STAMP Workshop, MIT, 2013

크로스 플랫폼 애플리케이션의 테스트 자동화 방안

황준선 ○ 최은만
동국대학교
hwangsun12@naver.com emchoi@dongguk.edu

A Method of Automated Testing Cross-Platform Application

요 약

애플리케이션을 개발할 때 소프트웨어가 요구사항을 만족하는지 확인하고 구현 후 결함을 찾기 위해서 테스트는 필수적이다. 크로스플랫폼 애플리케이션 제작 프레임워크인 일렉트론은 주로 스펙트론이라는 테스트 라이브러리를 모카라는 테스트 프레임워크에서 사용하여 테스트 코드를 작성하고 실행시킨다. 그러나 모카는 확장성, 로그와 리포트 생성, 테스트 케이스 작성과 관리 등 여러 가지 측면에서 로봇 프레임워크와 비교하였을 때 비효율적이다. 따라서 모카 프레임워크 대신 로봇 프레임워크를 사용하여 테스트 코드를 작성하면 테스트 효율성이 증가한다. 본 논문에서는 기존에 모카 프레임워크를 사용하던 일렉트론 애플리케이션 테스트를 로봇 프레임워크로 대체하여 효과적으로 테스트하는 방법을 제안하고, 로봇 프레임워크에서 지원하는 라이브러리와 가상화 도구를 사용하여 크로스 플랫폼 테스트를 더 효율적으로 하는 방안에 대해 제안한다.

1. 서 론

기존의 애플리케이션 개발은 각각의 플랫폼을 위한 코드를 작성하여 애플리케이션을 개발하는 것이 일반적이었다. 하지만, 이러한 방법으로 여러 플랫폼을 위한 애플리케이션을 만들면 개발비용, 보수비용이 플랫폼 개수만큼 늘어난다. 그래서 각각의 플랫폼을 위한 코드를 작성하는 것이 아니라, 하나의 코드로 모든 플랫폼에서 같은 동작을 하는 코드를 작성해 개발하는 방법이 주목을 받았다. 위와 같은 배경으로, 크로스 플랫폼 애플리케이션 개발 도구인 일렉트론이 나오게 되었다.

애플리케이션의 품질을 유지하기 위해서는 테스트를 해야 한다. 기존의 일반 애플리케이션을 테스트 할 때는 한 개의 플랫폼만 테스트하면 된다. 그러나 기존 애플리케이션과 다르게 일렉트론으로 만든 애플리케이션은 여러 플랫폼에서 동작한다. 따라서 일렉트론으로 만든 애플리케이션의 품질을 유지하기 위해서는 모든 플랫폼에서의 테스트가 필요하다.

일렉트론 테스트는 주로 Mocha라는 프레임워크로 테스트 코드를 작성하고 실행한다. Mocha는 기존 Node.js 애플리케이션을 테스트 할 때 적합하다. 그러나 크로스 플랫폼 애플리케이션을 테스트 할 때 각 플랫폼에서 테스트를 진행해야 한다. 즉, Mocha로 일렉트론 애플리케이션을 테스트하면(윈도우, 맥, Linux) 3개에 대한 3번의 테스트가 필요하다. 또한 Mocha는 자동화 테스트 기능을 제공하지 않아 애플리케이션을 효과적으로 테스트할 수 없다.

따라서 본 논문에서는 크로스 플랫폼 애플리케이션 테스트에 부적합한 Mocha 프레임워크를 Robot 프레임워크로 대체한다. Robot은 Mocha에서 지원하지 않는 높

은 확장성, GUI 기능, 자동화, 로그와 리포트 자동생성 등 여러 가지 장점이 있다. 그러나 Robot은 일렉트론 애플리케이션과 다른 프로그래밍 언어로 되어있어 테스트 대상과의 연결이 되지 않는 문제점이 있다.

문제점을 해결하기 위해 본 논문 3장에서 일렉트론 테스트를 위한 스펙트론 라이브러리를 서버파일로 구현한다. 구현한 서버 파일을 사용해 Robot과 테스트 대상을 연결하여 장점이 많은 Robot으로 테스트 하는 방법에 대해 설명한다. 또 테스트 대상과의 연결을 위해 만든 서버파일과 가상화 도구를 이용하여, Robot으로 여러 플랫폼을 한 번에 테스트하고 Robot의 기능을 사용하여 자동화 하여. 크로스 플랫폼 애플리케이션 자동화 테스트 방안에 대해 제안한다.

2. 관련 도구

크로스 플랫폼 애플리케이션을 개발하고 테스트 할 때 여러 가지 도구들이 사용된다. 또 본 논문에서 비교를 위해 사용되는 프레임워크와 대체 프레임워크, 애플리케이션 개발 도구, 애플리케이션 테스트 라이브러리 등 관련 도구들은 이번 장에서 설명한다.

2.1 노드 JS

노드 JS는 확장성 있는 네트워크 애플리케이션 개발에 사용되는 소프트웨어 플랫폼이다. 내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 별도의 소프트웨어 없이 동작하는 것이 가능하며 이를 통해 웹 서버의 동작에 있어 더 많은 통제를 가능하게 한다.

2.2 일렉트론

일렉트론은 데스크톱 애플리케이션을 만들기 위한 프레임워크이다. 주요 특징으로는 일렉트론은 웹 애플리케이션과 마찬가지로 JavaScript, HTML, CSS를 사용해서 애플리케이션을 만들 수 있다는 점이 있다. 또한 일렉트론으로 작성된 프로그램은 Windows, MacOS, Linux 모든 플랫폼에서 동작한다.

2.3 스펙트론

스펙트론은 일렉트론 애플리케이션을 E2E 테스트하기 위한 테스트 라이브러리다. 스펙트론을 사용하면 일렉트론 애플리케이션이 제공하는 API를 애플리케이션 외부에서 사용할 수 있게 된다. E2E 테스트는 애플리케이션을 실제로 조작해서 테스트하는 것이므로 외부에서 애플리케이션을 조작할 수 있는 도구가 필요하다.

2.4 VM웨어

VM웨어는 운영체제 가상화 도구이다. 한 개의 컴퓨터에서 소프트웨어 사용하면, 운영체제의 설치 없이 마이크로소프트 윈도우, 리눅스, OS X를 사용할 수 있다.

2.5 Mocha 프레임워크

Mocha 프레임워크는 노드JS에서 사용하는 테스트 프레임워크이다. 테스트 슈트와 테스트 케이스를 작성하고 작성한 테스트케이스를 실행하여 테스트 결과를 확인할 수 있는 기능을 제공한다.

표 1 로봇 프레임워크와 모카 프레임워크 비교표

	로봇 프레임워크	모카 프레임워크
장점	-간단한 테스트 케이스 작성 -읽기 쉽고 정보 많은 테스트 로그와 리포트 제공 -HTML형식의 로그와 리포트 자동생성으로 문서화 용이 -용이한 테스트 케이스 관리 -넓은 확장성 -GUI기능 제공 -자동화 테스트 지원 -키워드 기반 테스트	-비교적 세부적인 테스트 케이스 작성 가능 -노드JS 테스트에 특화되어있음
단점	-비교적 세부적인 테스트 불가	-낮은 확장성 -자동화 테스트 불가

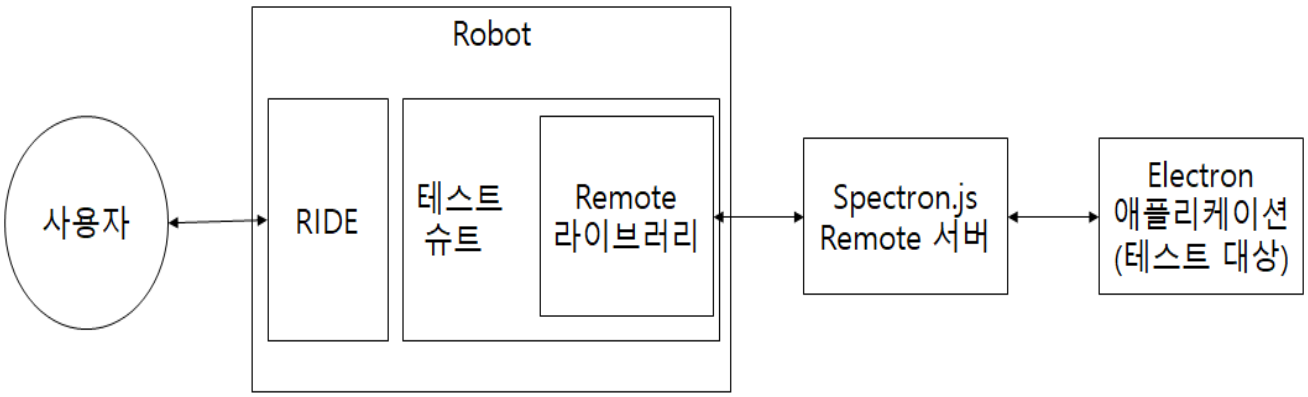


그림 1 로봇 프레임워크 테스트 개념도

2.6 Robot 프레임워크

Robot 프레임워크는 파이썬 기반 자동화 테스트 프레임워크이다. 테스트 케이스를 테이블 형식으로 쉽고 직관적으로 작성할 수 있다. 웹, 모바일, 데이터베이스 등 다양한 테스트 라이브러리를 제공해준다.

2.6.1 RIDE

Robot프레임워크를 GUI 환경으로 바꾸어주는 도구이다. 테스트 케이스 관리, 테스트 슈트 관리, 테스트 실행 등 여러 가지 기능들을 GUI 환경에서 실행할 수 있다.

3. Robot 프레임워크를 이용한 테스트 방안

2장의 표1 Robot과 Mocha의 비교에서 Robot 프레임워크의 장점으로 Mocha 프레임워크를 대체하여 Robot 프레임워크를 사용하는 방안에 대해 제안한다.

Mocha 프레임워크는 테스트 대상과 같은 프로그래밍 언어로 작성되어 그림 1과 같이 라이브러리를 추가하여 테스트 케이스를 작성하면 된다.

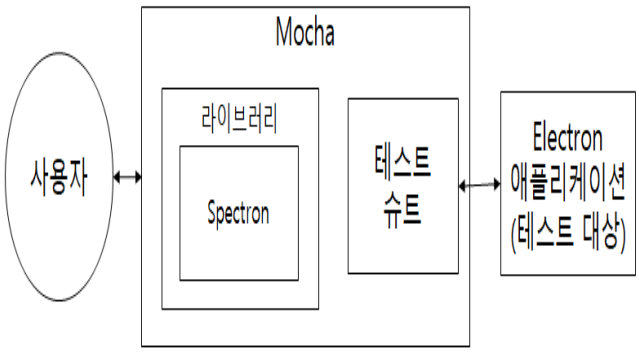


그림 2 기존 모카를 사용한 테스트 개념도

그러나 Robot 프레임워크는 파이썬으로 작성되어 테스트 라이브러리, 테스트 대상과 연결이 되지 않는다. 다른 언어로 작성된 테스트 대상과 연결하기 위하여 그림 2와 같이 Remote 라이브러리를 사용하여 테스트 대상과 연결하였다. 그림 2를 보면 Remote 라이브러리와 Remote 서버를 이용하여 테스트 대상과 연결한다. 이때 사용하는 Remote라이브러리에 대한 설명은 아래와 같다.

3.1 Remote 라이브러리

Remote 라이브러리는 Robot 프레임워크에서 기본적으로 지원하는 라이브러리 중 하나이다. Remote 라이브러리는 그림 3과 같이 Remote 서버를 사용하여 테스트 라이브러리와 테스트 대상을 Robot 프레임 워크와 연결시킨다. Remote 라이브러리를 사용하여 얻을 수 있는 이점은 아래와 같다.

1. Robot 프레임워크가 지원하지 않는 언어로 테스트 라이브러리를 작성할 수 있다.
2. Robot 프레임워크 자체가 실행되는 곳이 아닌 다른 기계에서 테스트 라이브러리를 실행 할 수 있다.

이번 장에서는 사용하는 이유 중 첫 번째인 다른 언어로 테스트 라이브러리를 작성하는 것을 적용한다.

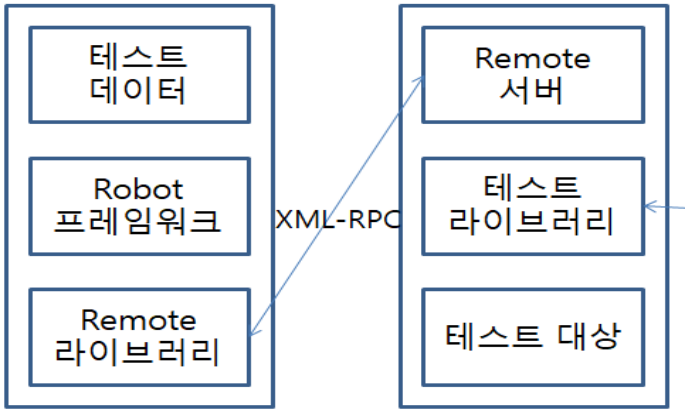


그림 3 Remote 라이브러리 개념도

3.2 Spectron.js 서버 구현

Robot 프레임워크에서 다른 언어로 테스트 라이브러리를 작성하기 위해서 Spectron.js 서버파일을 작성하였다. 그림 4와 같이 Spectron.js 서버파일은 2.3에서 설명한 Spectron이라는 테스트 라이브러리와, Electron 라이브러리, 3.1에서 설명한 Remote 라이브러리를 사용하여 구현한다. Spectron 기능 모듈화는 사용할 테스트 라이브러리 기능을 Robot프레임워크에서 사용할 수 있도록 함수로 만들어준다. 그림 5는 Spectron.js 서버파일을 일부 구현한 소스코드이다. 소스코드 4번째 줄은 Remote 라이브러리 추가, 소스코드 9번째 줄은 Spectron 라이브러리 추가, 소스코드 10번째 줄은 Electron 라이브러리의 추가이다. 13~28번째 줄은 표2의 스펙트론 라이브러리의 start(), stop()함수를 모듈화 시킨 부분이다. 애플리케이션 실행, 종료 기능을 startApp, stopApp 키워드로 구현하였다. startApp 키워드에는 일렉트론 애플리케이션의 경로를 찾고, start()함수를 사용하는 기능을 한다. 실제 일반적인 테스트에서 사용하기 위해서 표2의 스펙트론 라이브러리의 다른 함수들을 모두 모듈화 해야 한다. 나머지 함수는 향후 구현 예정이다. 31~34번째 줄은 Remote 라이브러리를 이용하여 서버를 열 수 있도록 구한한 소스코드이다.

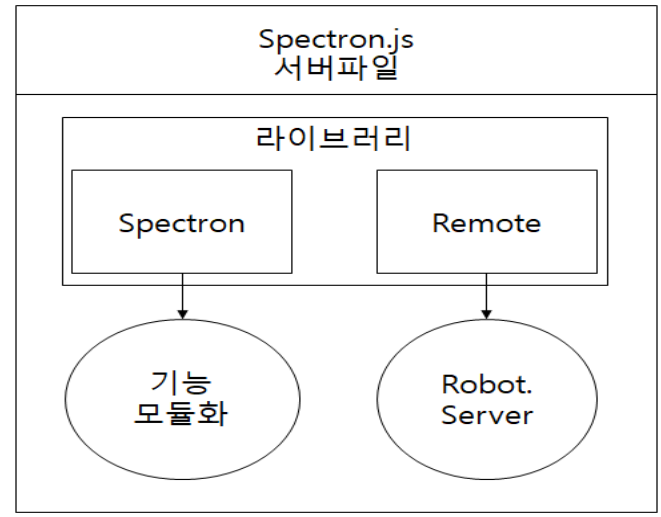


그림 5 Spectron.js 서버파일 개념도

```
1  'use strict';
2
3  var readdir = require('promise').denodeify(require('fs').readdir),
4      robot = require('robotremote/lib/robotremote'),
5      assert = require('assert');
6
7  var lib = module.exports;
8
9  const Application =require("spectron").Application;
10 const electron = require("electron");
11 const path = require("path");
12
13 lib.startApp = function () {
14    this.app = new Application({
15       path: electron,
16       args: [path.join(__dirname, '..')]
17    })
18    return this.app.start()
19 };
20
21 lib.startApp.doc = 'Returns the number of items in the directory specified by `path`.';
22
23
24 lib.stopApp = function () {
25    if (this.app && this.app.isRunning()){
26       return this.app.stop()
27    }
28 };
29
30
31 // Run this keyword library if the library itself is called explicitly.
32 if (!module.parent) {
33    var server = new robot.Server([lib], { host: 'localhost', port: 8270 });
34 }
```

그림 4 Spectron.js 서버파일 일부 구현

표 2 스펙트론 라이브러리 함수

함수	설명
Start()	응용 프로그램을 실행한다.
Stop()	응용 프로그램을 종료한다.
Restart()	응용 프로그램을 재시작한다.
isRunning()	응용 프로그램이 실행중인지 확인한다.
getSettings()	응용 프로그램의 모든 옵션을 가져온다.
Client.getMainProcessLogs()	주 프로세스에서 로그 출력을 가져온다.
client.getRenderPro	랜더링 프로세스에서 로그 출력을 가

cessLogs()	져온다.
client.getSelectedText()	현재 창에서 선택한 텍스트를 가져온다.
client.getWindowCount()	열려있는 창의 수를 가져온다.
client.waitUntilTextExists(selector, text,[timeout])	지정된 selector와 일치하는 요소가 지정된 텍스트를 포함 할 때까지 대기한다.
client.waitUntilWindowLoaded([timeout])	창이 더이상 로드되지 않을 때까지 대기한다.
client.windowByIndex(index)	배열의 색인을 사용하여 창에 초점을 맞춘다.

3.3 Robot 프레임워크와 테스트 대상 연결

3.2장에서 구현한 서버파일을 사용하여 Robot 프레임워크와 테스트 대상인 일렉트론 애플리케이션을 연결한다. 테스트 대상과 연결하는 과정은 아래와 같다.

1. NPM 명령어를 사용하여 3.1장에서 설명한 Remote 라이브러리를 다운로드 받는다.
2. Robot 프레임워크에서 Remote라이브러리를 추가하고 호스트와 포트번호를 입력한다.
3. Node 명령어를 사용하여 그림 6과 같이 Remote 서버를 실행시킨다.
4. Robot프레임워크에서 Remote라이브러리의 색깔로 연결 되었는지 확인한다. 검은색이면 연결 성공, 빨간색이면 연결 실패이다.

그림 6 Spectron.js 서버파일 실행

3.4 RIDE를 사용한 테스트 케이스 작성

Robot 프레임워크와 서버파일이 정상적으로 연결이 되면 그림 7과 같이 모듈화 시킨 테스트 키워드를 사용할 수 있다.

2장에서 설명한 GUI 변환도구인 RIDE를 사용하여 테스트 케이스를 작성한다. RIDE 사용하였을 때 Mocha에서 지원하지 않는 기능을 사용할 수 있다. RIDE에서만 지원하는 기능은 아래와 같다.

-그림 7과 같이 테이블 형식의 쉬운 문법으로 테스트 케이스를 작성할 수 있다.

-테이블 형식으로 만든 테스트 케이스를 키워드로 만들어 쉽게 재사용 할 수 있다.

-GUI 환경에서 테스트 슈트, 테스트 케이스를 한눈에 볼 수 있어 쉽게 테스트를 관리할 수 있다.

그림 8은 Robot 프레임워크와 Mocha 프레임워크로 작성된 테스트 케이스 비교이다. 두 테스트 케이스 모두

그림 7 로봇 프레임워크 환경에서 테스트 케이스 작성

아래와 같은 동작을 한다.

1. 애플리케이션을 실행시킨다.
2. 애플리케이션이 1개만 실행되었는지 확인한다.
3. 애플리케이션을 종료시킨다.

같은 동작을 하는 테스트 케이스이지만, Robot 프레임워크를 사용한 테스트 케이스는 3줄 Mocha 프레임워크로 작성된 테스트 케이스는 29줄이다.Robot 프레임워크를 사용하였을 때 같은 동작을 약 10배 적은 소스 코드로 작성하였다. Robot 프레임워크를 사용하게 되면 테

스트 케이스 작성 시간 절약으로 테스트 비용을 획기적으로 감소시킬 수 있게 된다

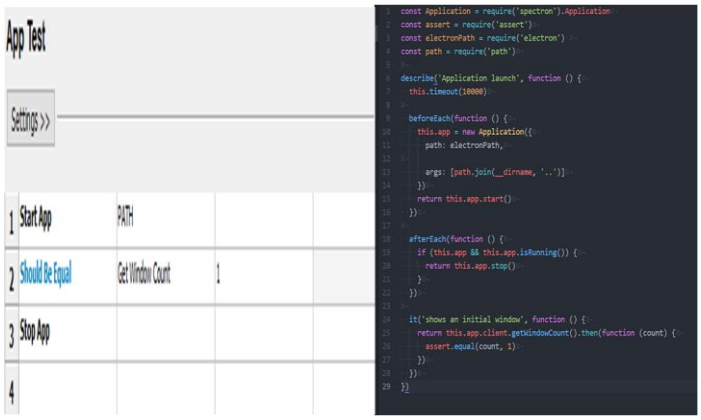


그림 8 로봇 프레임워크, 모카 테스트 케이스 비교

3.5 테스트 코드 실행

Robot 프레임워크로 작성한 테스트 케이스는 GUI 환경에서 실행 버튼을 클릭하면 실행된다. 테스트를 실행하게 되면 테스트 로그와 테스트 리포트 파일이 HTML 파일 형식으로 자동 생성된다. 생성된 로그와 리포트에는 테스트 성공, 실패, 실행시간, 시작시간, 종료시간 등 테스트와 관련된 정보들이 있다. Robot 프레임워크가 자동으로 생성한 로그와 리포트는 Mocha 프레임워크의 테스트 로그와 비교하였을 때 아래와 같은 장점이 있다.

1. 가독성이 좋고 더 많은 정보를 상세하게 볼 수 있다.
2. HTML 파일로 자동 생성되어 문서화에 용이하다

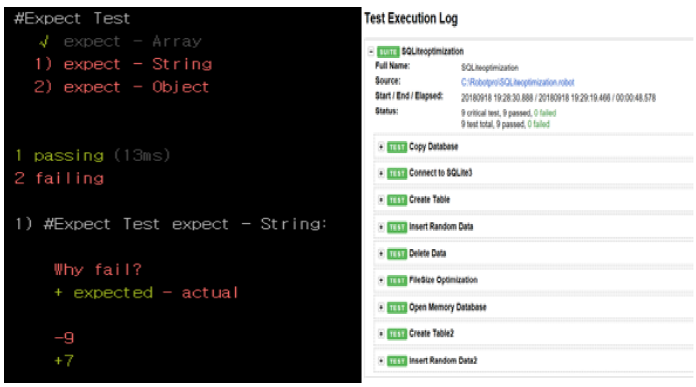


그림 9 모카, 로봇 프레임워크 테스트 로그 비교

4.테스트 자동화 방안

3장에서는 모카보다 장점이 많은 로봇 프레임워크를 사용하여 일렉트론 애플리케이션을 테스트하는 방안에 대해 제안했다. 이번 장에서는 로봇 프레임워크에서 사용 가능한 자동화 기능과 자동화 방안에 대해 설명하겠다.

4.1 회귀 테스트

소프트웨어 유지보수, 기능의 확장 등 버전관리를 하는 프로그램의 경우 회귀 테스트를 한다. 회귀 테스트는 이전에 정상적으로 작동된 프로그램에 일어나는 버그, 즉 회귀 버그를 발견하기 위한 테스트이다. 회귀테스트는 요구사항이 변경될 때 마다 기존 기능의 영향여부를 확인하기 위해 실행한다. 회귀테스트를 진행할 때는 기존 테스트에 사용하였던 테스트 케이스를 사용하여 테스트를 진행한다. 회귀 테스트를 하는 이유는 아래와 같다.

-소프트웨어 복잡성으로 기존의 수정된 버그가 소프트웨어 변경에 의해 재 발생할 가능성 방지.
-기존 버그수정을 위한 소프트웨어 변경으로 예상하지 못한 버그 발생 방지.
-결함조치 확인 및 운영상의 일부 모듈의 변경에 따른 전체적인 정합성 재확인.

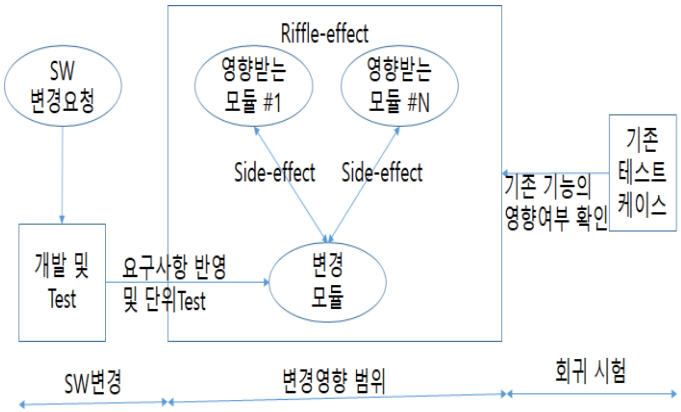


그림 10 회귀 테스트 개념도

4.2 로봇 프레임워크 자동화 필요성

앞장에서 회귀테스트의 필요성과 개념에 대해 설명하였다. 그러나 모카 프레임워크는 테스트 실행 자동화 기능이 없어 회귀 테스트를 하기 어렵다. 예를 들어 기존 테

스트 케이스가 100개가 존재한다면, 요구사항 변경마다 각 플랫폼 별 회귀 테스트를 하기 위해 300번 이상의 테스트를 일일이 실행해 주어야 한다. 또한 각 테스트 결과인 테스트 로그를 테스터가 직접 통합해 주어야 한다. 따라서 모카 프레임워크로 회귀 테스트를 하는 것은 비효율적이며, 비용이 너무 높아 사실상 불가능하다. 로봇 프레임워크를 사용할 경우 회귀 테스트 자동화 기능을 제공한다. 그림 11과 같이 지금까지 수행한 테스트 케이스가 저장되어 있어 필요한 테스트 케이스만 체크하여 실행하거나 테스트 케이스를 모두 일괄적으로 실행하고 실행한 테스트를 그림 9와 같이 자동으로 생성된 로그에서 일괄적으로 테스트 결과를 확인할 수 있다.

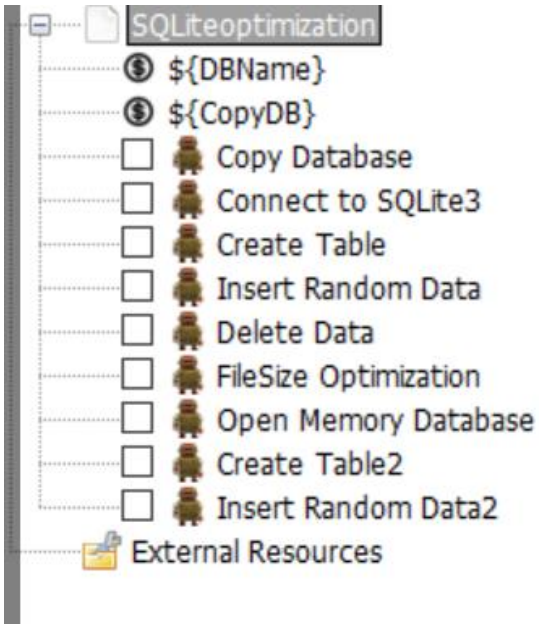


그림 11 로봇 프레임워크 테스트 케이스 목록

5.크로스 플랫폼 테스트 방안

본 논문의 테스트 대상은 1장에서 설명한 것처럼 크로스 플랫폼 애플리케이션이다. 일렉트론으로 제작된 애플리케이션의 경우 Window, Linux, X(맥 OS) 등 세 가지 OS에서 도작한다. 모카 프레임워크의 경우에는 각각의 테스트 환경에서 모두 테스트를 진행해야 했다. 이번 장에서는 3장에서 만든 스펙트론 서버파일을 응용하여 3가지 OS를 모두 한번에 테스트 하는 방안에 대해 제안한다.

5.1 기존 크로스 플랫폼 테스트

기존에 크로스 플랫폼 테스트를 하기 위해 그림 12와 같이 각 플랫폼을 테스트하기 위해서 테스트 프레임워크와 애플리케이션이 설치된 OS별 기기가 있어야 했다. 모카 프레임워크, 로봇 프레임워크 모두 애플리케이션이 설치된 OS별 기기가 있어야 한다. 기존의 경우에는 테스트를 하기 위해 아래와 같은 과정을 거쳤다.

1. 테스트 케이스를 작성한다.
2. 각 OS에 작성한 테스트 케이스를 복사한다.
3. 각 OS에 애플리케이션을 업데이트 한다.
4. 각 OS별로 테스트 케이스를 실행한다.
5. 각 OS에서 생성된 테스트 로그를 통합한다.

매번 요구사항이 변경될 때 마다 위와 같은 과정을 거치기에 크로스 플랫폼 애플리케이션 테스트는 일반 애플리케이션과 비교하였을 때 테스트 비용이 높았다.

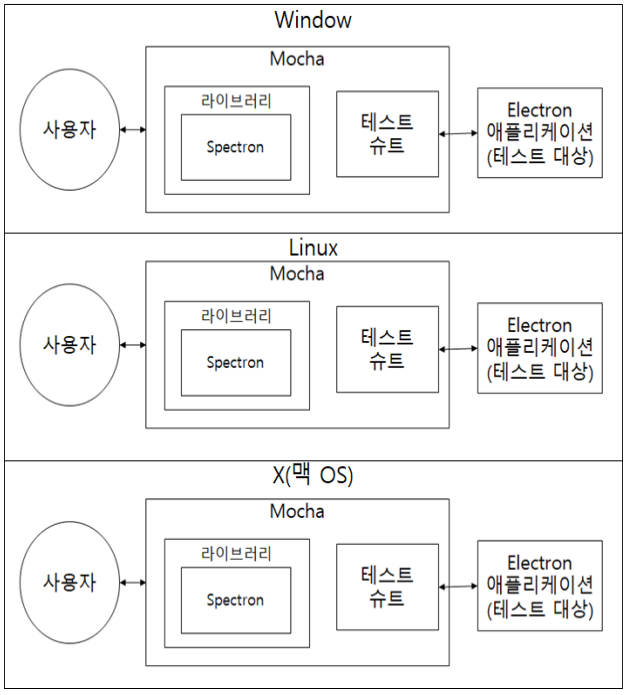


그림 12 기존 크로스플랫폼 테스트 개념도

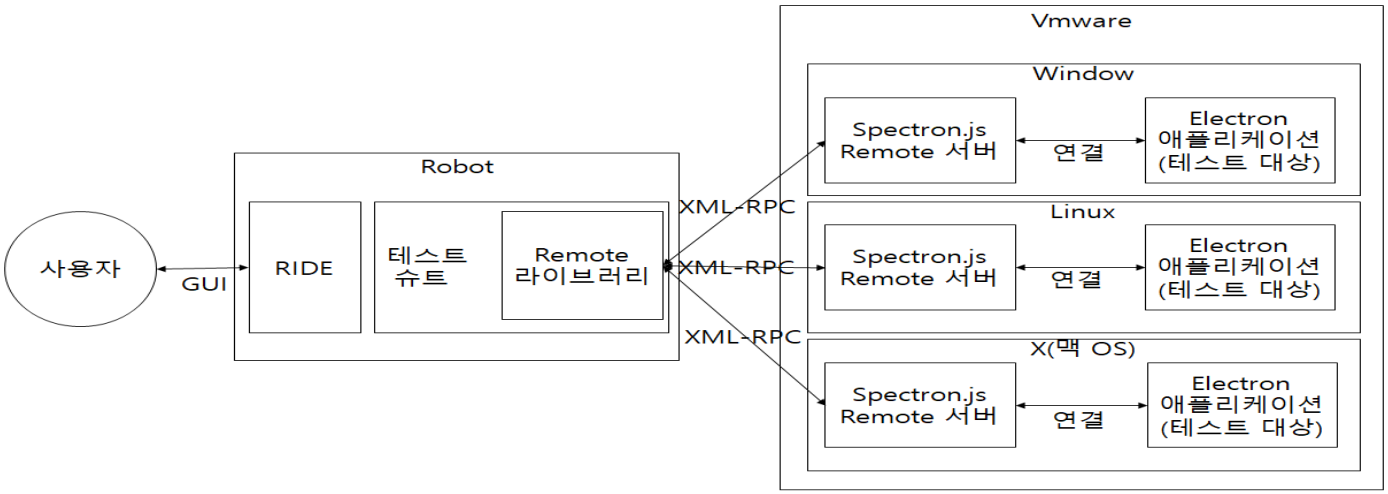


그림 13 로봇 프레임워크 크로스플랫폼 테스트 개념도

5.2 크로스 플랫폼 일괄 테스트 방안

앞장에서 기존의 크로스 플랫폼 테스트를 할 때 일반 테스트와 비교하였을 때 복잡함을 설명하였다. 이번 장에서는 복잡했던 크로스 플랫폼 테스트를 일괄적으로 하는 방안에 대해 자세히 설명한다. 그림 13은 3장에서 만든 서버파일과 VmWare를 사용하여 여러 플랫폼을 일괄적으로 하는 개념도이다.

3장에서 만든 서버파일을 사용하여 각 서버파일을 OS별로 넣어주어 테스트 대상과 연결하여 애플리케이션을 일괄적으로 테스트한다. 서버파일에서 사용한 Remote 라이브러리를 사용하여 얻는 이점은 아래와 같다.

1. Robot 프레임워크가 지원하지 않는 언어로 테스트 라이브러리를 작성할 수 있다.
2. Robot 프레임워크 자체가 실행되는 곳이 아닌 다른 기계에서 테스트 라이브러리를 실행 할 수 있다.

3장에서는 첫 번째 이점을 사용하여 모카를 로봇 프레임워크로 대체하였다. 이번 장에서는 두 번째 이점을 사용하여 크로스 플랫폼 애플리케이션 테스트 방안을 제안한다. 두 번째 이점의 경우 본래 분산 환경에서의 테스트를 위해 사용한다. 본 논문에서는 두 번째 이점을 분산 테스트 대신 크로스 플랫폼 애플리케이션 테스트에 적용한다. 이를 사용하여 다른 플랫폼에 있는 테스트 대상과 로봇 프레임워크를 연결할 수 있게 된다. 로봇 프레임워크 환경은 3장에서 설명한 그림 2와 동일하다. 크로스 플랫폼 일괄 테스트 과정은 아래와 같다.

1. 2장에서 설명한 Vmware를 사용하여 Window, Linux, X(맥 OS) 가상화를 실행 시킨다.
2. 각 OS에 테스트 대상 애플리케이션을 설치한다.
3. 각 OS에서 Remote 서버를 실행시킨다.
4. 로봇과 Remote 서버가 연결되었는지 확인한다.
5. 작성한 테스트케이스를 실행한다.

6. 결론 및 향후 연구내용

본 논문은 로봇 프레임워크와 모카 프레임워크를 비교하여 더 효율적인 테스트가 가능한 로봇 프레임워크를 소개, 기능 개선, 기능 확장을 하였다.

3장에서는 Remote 라이브러리와 스펙트론 라이브러리를 사용하여 직접 만든 서버 파일로 로봇 프레임워크에서 불가능하였던 일렉트론 애플리케이션을 테스트 할 수 있게 기여하였다.

4장에서는 데스크톱 애플리케이션 특성상 규모가 모바일 애플리케이션보다 큰 경우가 많아 모듈끼리의 간섭이 많다. 모듈간의 간섭으로 소프트웨어 변경으로 기존

에 없던 새로운 오류가 생길 수 있다. 따라서 로봇프레임워크의 대표적인 장점인 자동화 기능을 사용하여 회귀테스트를 진행하는 방안에 대하여 제안하였다.

5장에서는 기존에 분산 테스트를 위한 기능을 크로스 플랫폼에 애플리케이션 테스트에 적용할 수 있다는 가능성과 방법에 대해 기여하였다. 제안 방안으로 3개의 플랫폼을 동시에 테스트 할 수 있게 하여, 테스트 기기 수와, 테스트 횟수를 3배 이상 감소시켰다.

향후 연구로는 표2에 나와 있는 스펙트론 라이브러리의 나머지 함수들을 모두 구현하여 실제 테스트에서 사용할 수 있도록 만들고 배포할 것이다. 또한 3장에서 만든 서버파일은 일렉트론과 연결하기 위해 일렉트론 라이브러리도 포함된다. 이 점을 이용하여 서버파일에 애플리케이션 자동 업데이트 기능을 추가하여 서버파일의 효용을 올리고 테스트를 더 효율적으로 할 수 있도록 연구할 것이다.

6. 참고 문헌

[1] N. Masato, C. Yoseke, *Cross-Platform Application Development*, Freelec, 2017.

[2] S. Stresnjak, Z. Hocenski, “Usage of Robot Framework in Automation of Functional Test Regression”, ICSEA, pp 30-34, 2011.

[3] Robot Framework, "http://robotframework.org/"

[4] Node.js, "https://nodejs.org/"

[5] Electron, "https://electronjs.org/“

[6] Mocha Framework, “https://mochajs.org/”

[7] Robot Framework Node Connection Library, “https://github.com/comick/node-robotremoteserver”

[8] Robot Framework Node Test Runner, “https://github.com/samick17/node-robotframework-test-runner”

Engine Control Unit (ECU) 실시간 객체 지향 모델의 능동 객체 기반 우선 순위 전파 멀티 코어 멀티 태스킹*

김경률 ○, 김세화

한국외국어대학교 정보통신공학과

kyoung831@naver.com, ksaehwa@hufs.ac.kr

Active Object-Based and Priority Propagated Multicore Multitasking of Engine Control Unit (ECU) Real-Time Object-Oriented Model

KYOUNGRUL KIM○ and SAEHWA KIM

Dept. of Information Communications Engineering, Hankuk Univ. of Foreign Studies

요 약

본 논문에서는 Engine Control Unit (ECU) 실시간 객체지향 모델을 멀티 코어 시스템에서 효율적으로 수행시키기 위한 방법을 논한다. 우선 능동 객체를 태스크로 매핑하고, 태스크를 슬랙을 기반으로 멀티 코어에 매핑한다. 각 태스크는 고정 우선순위를 가지는 것이 아니라, 수행하는 시나리오에 따라 우선순위가 변동된다. 우리는 이를 우선순위 전파 기법이라 명명한다. 시나리오의 응답시간을 이론적으로 분석할 뿐 아니라, 오픈 소스 객체 지향 모델링 도구 eTrice를 기반으로 Linux 환경에서 객체 지향 모델링을 통하여 ECU 시스템을 구현하여 그 결과를 분석한다. 이를 통해 우선순위 전파 기법이 시나리오의 응답 시간을 이론적, 실험적으로 모두 향상시키는 것을 보인다.

1. 서 론

최근 자율주행자동차, IoT등 우리 생활에 가까이 있는 다양한 임베디드 시스템이 점점 복잡해지고 있다. 이와 동시에 임베디드 시스템에 탑재되는 소프트웨어의 복잡성 역시 높아지고 있다. 이로 인해 소프트웨어의 개발과 수정 등의 유연성을 향상시킬 필요성 역시 높아졌다. 그래서 임베디드 시스템 소프트웨어 개발을 모델링을 기반으로 수행하는 것이 필연적이게 되었다.

소프트웨어 모델링 언어인 UML(Unified Modeling Language)은 UML2.0 이후 실시간 객체 지향 모델(Real-Time Object-Oriented Model, ROOM)이 포함되면서 실시간 응답성이 중요한 임베디드 시스템에서 산업계 실질적인 표준 모델링 언어가 되었다. 이에 따라 UML을 이용하여 실시간 객체 지향 모델링을 설계하고, 설계한 모델에 기반하여 실행 가능한 코드를 자동 생성해 주는 도구 역시 계속해서 개발되고 있다. 이러한 도구로써 IBM사의 RoseRT[1], Telelogic I-Logix Rhapsody[2]가 있고, 오픈 소스로서 eTrice[3]등이 있다.

임베디드 시스템은 저전력으로 높은 성능과 복잡한 기능 요구사항들을 만족하기 위해 클락 주파수를 낮출 필요성이 있다. 그리고 하드웨어의 성능이 급격히 좋아졌을 뿐 아니라 가격마저 저렴해 졌기 때문에 점점 더 많은 임베디드 시스템이 싱글 코어가 아닌 멀티 코어를 기반으로 설계되고 있다. 그러나 모델링 도구에서는 시스템이 가진 태스크들을 멀티 코어로 매핑하는 것에 대한 구체적인 기능이 부족한 단계이며, 해당 연구 또한 부족하다.

AUTOSAR는 소프트웨어와 하드웨어를 분리하는 기본 아이디어를 바탕으로, 표준화 된 인터페이스를 사용하여 소프트웨어를 독립적인 구성요소로 모듈화 한다는 아이디어를 가지고 있다. 그러나 AUTOSAR는 Engine Control Unit(ECU)에서 소프트웨어를 통합하게 되면서 생기는 태스크 복잡성에 관해 다루지 않기 때문에

* 이 논문은 2017 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2017R1A2B1001824).

ECU시스템을 사례로 연구하였다.

본 논문에서는 현재까지 개발된 다양한 모델링 도구들 중에서 오픈 소스인 eTrice를 사용한다. 그리고 eTrice에서 자동차의 ECU시스템을 설계하고 구현한다. 이 과정에서 eTrice에 포함되어 있지만 제대로 수행되지 않는 기능인 스레드 우선순위 설정 기능을 수정하고, 포함되어 있지 않은 다양한 스레드를 특정 코어에 할당할 수 있는 기능을 추가하여 구현하며, 시나리오에 따른 스레드의 우선순위 전파 기능 또한 추가로 구현하였다. 그리고 구현된 기능들을 이용하여 여러 방법으로 태스크를 코어에 매핑하고 우선순위를 설정한다. 이를 토대로 임베디드 시스템의 최적 설계가 어떠한 환경에서 찾을 수 있는 지 연구한다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 2장에서는 시스템 모델과 가정 그리고 eTrice 모델링 도구와 기능에 대해 설명한다. 그리고 3장에서는 ECU 시스템 모델 설계와 모델에 대한 시나리오 구성, 능동 객체들의 태스크 매핑에 대해 설명한다. 4장에서는 능동 객체 기반 멀티 태스킹에서의 우선순위 전파에 대해 설명하고 5장에서는 멀티 코어 매핑과 이론적 분석에 대해 설명한다. 6장에서는 eTrice에서의 실험적 결과에 대해 분석하고, 마지막으로 7장에서는 결론을 맺는다.

2. 시스템 모델 가정과 도구

2.1 시스템 모델과 가정

본 논문에서 실시간 객체 지향 모델과 태스크 모델에 대한 가정은 [5]을 따른다. 실시간 객체 지향 모델[6]에서 각 객체의 상태 천이는 완전 수행 의미 체계(run-to-completion semantics)를 따르며, 각 상태 천이는 최악 수행 시간(Worst-case Execution Time, WCET)을 사용한다. 각 상태 천이는 주기적으로 수행될 수도 있고, 그렇지 않을 수도 있다.

본 논문에서는 멀티 코어 스케줄링 기법으로서, 전역 스케줄링이 아니라, 코어 별로 태스크들이 미리 할당된 분할된(partitioned) 완전 선점 고정 우선 순위 스케줄링 방식을 채택한다[7]. 각 태스크 τ_i는 고정 우선순위 π_i, WCET C_i를 가지고, 주기 T_i를 가지며, 종료시한 D_i 는 주기와 같다고 가정한다($D_i = T_i$). 우선순위는 DMPO(Deadline Monotonic Priority Ordering)방법으로 할당한다. 즉, 종료 시한이 짧을수록 더 높은 우선순위를 할당 받는다. 또한 우선순위는 큰 숫자가 더 높은 우선순위를 나타낸다.

eTrice에서는 단순한 뮤텍스를 사용하고 있고, 이 뮤텍스에 의한 블로킹 타임이 상태 천이 시간과 완전 수행 의미 체계에 의한 블로킹 타임에 비해 현저히 작기 때문에 본 논문에서는 이론적 분석에서 이러한 뮤텍스에 의한 블로킹 타임은 무시한다. 또한 멀티 코어에서의 동기화 정책으로서는 같은 코어에 매핑된 태스크 간에 공유되는 자원(지역 공유 자원)에 대해서는 IIP[8]를 사용하고, 다른 코어에 매핑된 태스크 간에 공유되는 자원(전역 공유 자원)에 대해서는 가능한 경우 CAS(compare and swap)에 기반한 락 없는 동기화 기법을 사용한다고 가정한다.

태스크의 멀티 코어로의 매핑 기법은 GS-WF(Greedy Slacker-Wait Free)[9]를 기본으로 사용한다. GS-WF는 메모리 소모량을 고려하지 않은 상태에서 태스크들의 밀도 순에 따라 각 코어에서의 최소 슬랙(slack)을 최대화 하도록 태스크들을 코어에 할당하는 방식이다. 여기에서 각 태스크의 밀도는 WCET(Worst-Case Execution Time, 최악 수행 시간)를 종료시한으로 나눈 값으로 정의된다. 본 논문에서는 각 태스크의 종료시한과 주기가 같기 때문에 태스크 밀도가 태스크 이용률 Ui와 동일하다. 슬랙(slack)은 정규화된 슬랙으로서 각 태스크(τ_i)에 대해 그 태스크의 종료시한(D_i)과 최악 응답 시간(R_i)의 차이를 종료시한으로 나눈 값으로 정해진다($slack_i = (D_i - R_i)/ D_i)$.

시나리오는 외부 이벤트로부터 시작하여 외부 이벤트를 발생시키거나 시스템 내부의 변화를 발생시키는 일련의 메시지 처리 과정을 의미하며, 종료시한을 가지는 단위이다. 실시간 객체 지향 모델로 설계된 시스템의 응답 시간은 시나리오들의 응답 시간을 의미한다. 시나리오 그룹[10]은 특정 액터에 도착한 같은 종류의 외부 이벤트에서 시작한 모든 시나리오들의 묶음이다.

본 논문에서는 태스크의 개수를 시나리오의 그룹의 개수로 정한다. 객체가 시나리오 그룹의 개수 보다 많은 경우는 객체를 그룹 지어 태스크에 매핑시켜 시나리오 그룹 개수와 같게 만든다. 객체가 시나리오 그룹보다 적은 경우는 하나의 객체가 여러 주기성을 갖게 구현해야 하는데, 이는 비합리적이기에 이런 경우는 없다고 가정한다.

또한 본 논문에서는 우선순위가 [10][11]에서와 같이 시나리오 단위로 설정된다고 가정한다. 시나리오의 중간 액션이 매핑된 태스크가 시작 노드와 다른

노드에서 실행된다면, 이때는 그 태스크로 시나리오의 우선순위를 전달 해야 할 필요성이 있다. 그렇기에 우선순위를 전달할 경우와 전달하지 않을 경우의 성능 비교를 수행한다. eTrice에서는 메시지에 우선순위 정보를 넣어 메시지를 처리하는 상태 천이에서 태스크의 우선순위를 동적으로 설정하는 것이 불가능하다. 그렇기 때문에 우선순위를 동적으로 변경할 수 있도록 기능을 추가적으로 구현하였다.

2.2 eTrice 실시간 객체 지향 모델링 도구

현존하는 실시간 객체 지향 모델링 도구는 임베디드 시스템을 텍스트가 아닌 그래픽 환경에서 모델링을 할 수 있을 뿐 아니라 모델링에 따른 코드를 생성해주는 기능까지 있기 때문에 대부분 가격이 비싸다. 하지만 eTrice의 경우 현존하는 실시간 객체 지향 모델링 도구 중 유일하게 무료 오픈 소스로 제공된다. 의료기기를 비롯한 각종 임베디드 시스템 장비를 만드는 protos사[4]에서 최초 제안하였으며 eclipse 플러그인으로 개발되었다.

eTrice는 다른 모델링 도구와 다를 바 없이 구조 모델 다이어그램과 행위 모델 다이어그램을 그래픽 환경을 이용하여 그림 1의 왼쪽과 같이 모델링을 하고, 동시에 그림 1 의 오른쪽 .room파일과 같이 텍스트 형식으로 저장해준다.

텍스트형식 파일에서는 액터 간의 포트를 바인딩하고 통신하는데 필요한 프로토콜을 정의할 수 있다. 정의된 프로토콜을 사용하여 액터들은 이벤트 메시지를 주고 받아 연결할 수 있다. 행위 모델 다이어그램에서는 포트를 통해 들어온 이벤트 메시지를 트리거로 설정하며, 트리거에서 이에 대응되는 행동을

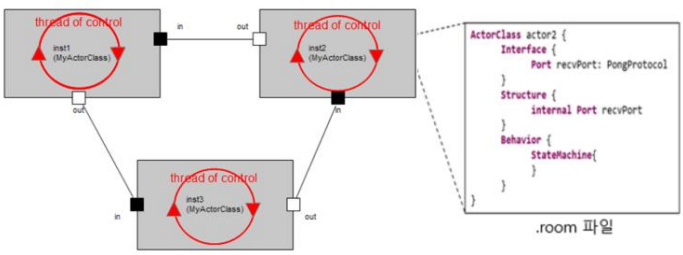


그림 1. eTrice ROOM 구성 : 회색사각형은 능동 객체(액터)를 나타내며 액터 테두리의 작은 검정색 사각형과 흰색 사각형은 각각 In, Out 포트를 나타낸다. 액터들은 매핑 된 스레드로 제어할 수 있다. 왼쪽과 같은 모델링을 하면 자동으로 오른쪽과 같은 텍스트 형식으로 .room파일이 생성된다.

정의함으로써 상세 수준의 코드를 구현하게 된다.

구조 모델 다이어그램과 행위 모델 다이어그램들을 통하여 구성된 파일은 .room파일이며, 이 .room파일에서는 다이어그램에 대한 것 뿐 아니라 프로토콜, 논리적 스레드 선언한다. 또한 선언된 논리적 스레드와 구조 모델 다이어그램을 통해 생성된 액터를 서로 매핑시킬 수 있다. .etphys 파일은 물리적 스레드의 실행모드, 우선순위와 같은 속성을 정의할 수 있다. .etmap 파일은 .etphys의 물리적 스레드와 .room파일의 논리적 스레드를 연결하여 액터를 결국 물리적 스레드에 매핑시키는 역할을 한다. 그리고 이 3가지 파일들을 기반으로 해서 실질적으로 수행될 수 있는 코드(C, C++, Java)가 생성된다.

본 논문에서는 사례연구를 위하여 .etphys파일의 우선순위 속성에 제대로 설정되지 않는 문제를 수정하였고, 스레드를 특정 코어로 할당할 수 있는 기능을 추가하였다. 그리고 사례연구가 시나리오 단위로 진행되기 때문에 시나리오의 우선순위를 태스크에게 전달하여 태스크의 우선순위를 변경할 수 있게 하였다. 또한 eTrice는 Timer의 Timing resolution 기본 값이 100ms로 설정되어 있지만 명시되어 있지 않다. 그렇기 때문에 100ms 보다 작은 Timer를 사용할 경우 기본 값을 수정해야하는 것에 주의해야 한다. eTrice의 nodeSubSystem.c에서 interval 정보를 수정하여 Timing resolustion을 변경할 수 있다. 본 연구에서는 가장 작은 주기인 5ms로 interval 정보를 낮추어 설정하였다.

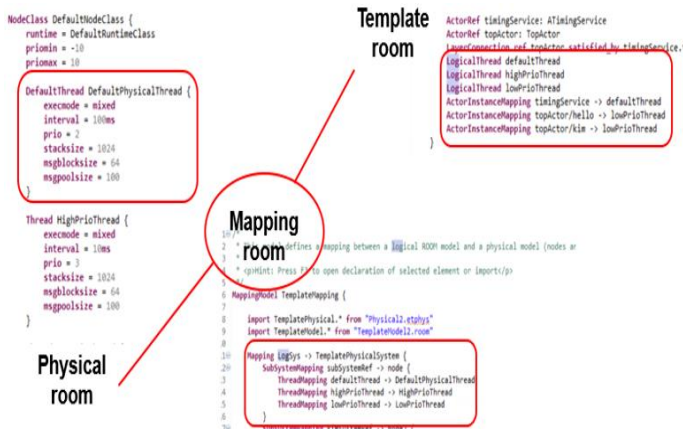


그림 2. eTrice의 Thread 구성 : eTrice는 .room파일 뿐 아니라 .etmap파일, .etphys파일들을 더하여 구성된다. 여기서 Template room은 .room파일을 나타내며, Mapping room은 .etmap파일, Physical room은 .etphys파일을 나타낸다.

3. ECU 시스템 모델

2.2장에서 eTrice에 대해 설명한 바와 같이 본 논문에서는 구조 모델 다이어그램과 행위 모델 다이어그램으로 AUTOSAR의 ECU 시스템 모델을 설계할 수 있다. 이를 위해 AUTOSAR runnable을 eTrice의 행위 모델 다이어그램에서 액션으로, AUTOSAR SWC를 eTrice의 구조 모델 다이어그램에서 액터(능동 객체)로 적용시켰다. 우리는 ECU 시스템 모델을 설계하기 위해 [12]과 [13], [14]를 참조하여 구조를 크게 기준을 4가지로 나누었다.

현재 스로틀 위치와 가속 페달 위치를 기준으로 원하는 스로틀 위치를 계산하여 공기 흐름에 영향을 미치는 AirSystem, PedalSensor의 진단 정보를 체크하는 PedalSensorDiag, 실린더 내 연소 과정을 위해 intake maniflod에 주입될 연료량을 계산하는 FuelingSystem, 연소실의 공기와 연료 혼합물을 점화하기 위해 점화 스파크가 발생할 시점을 결정하는 점화각도를 계산하는 IgnitionSystem으로 기준을 나누어 기준들과 관련된 액터들을 연관지어 설계하였다.

그림 3, 4, 5, 6은 각각 AirSystem, PedalSensorDiag, FuelingSysyem, IgnitionSystem과 연관된 액터들을 묶어 표현한 것으로 사각형은 각 시스템의 액터, 타원은 행위 상태, ■ 는 Reference Port(RP), □ 는 Conjugated Reference Port(CRP)를 나타낸다. 각 그림은 편의상 병합하여 나타낸 것이며 실제로는 구조 모델 다이어그램과 행위 모델 다이어그램은 따로 설계하여야 한다.

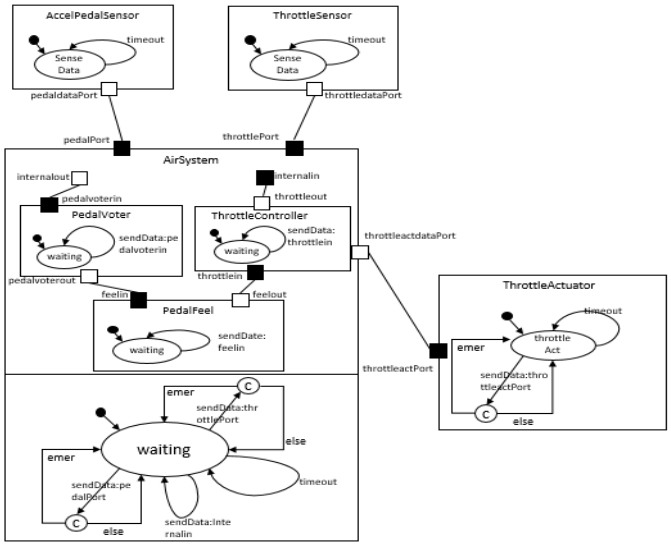


그림 3. ECU 시스템 중 AirSystem에 관련된 액터, 포트, 구조 모델 다이어그램

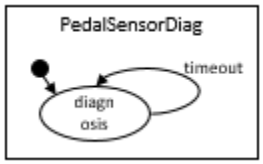


그림 4. ECU 시스템 중 PedalSensorDiag에 관련된 액터, 포트, 구조 모델 다이어그램

그림 3에서 AccelPedalSensor와 ThrottleSensor는 각각 가속페달의 위치와 스로틀 위치를 감지한다. 그리고 감지한 데이터를 timeout 트리거가 수행되면 각각의 외부CRP로 메세지를 보내서 AirSystem의 외부 RP에 전달한다.

AirSystem은 외부 RP, 내부 RP를 통해 메시지를 전달받아 저장한다. 또한 내부 CRP와 외부 CRP를 사용하여 메시지를 전달한다. 또한 AirSystem은 PedalVoter, ThrottleController, PedalFeel을 포함한 SubSystem을 포함하고 있다. Timeout 트리거가 발생되면 내부 CRP를 통해 SubSystem으로 메시지를 전달하고 SubSystem은 이를 처리하여 메시지를 AirSystem의 내부RP로 전송한다. 그리고 AirSystem은 메시지를 저장하고 이 데이터를 외부CRP를 통해 ThrottleActuator의 외부 RP에 전달한다.

ThrottleActuator는 외부RP를 통해 전달받은 데이터를 저장한다. 그리고 timeout 이벤트가 발생하면 저장된 데이터를 처리한다.

그림 4는 PedalSensorDiag에 관련된 다이어그램이며, PedalSensorDiag는 페달 센서를 진단하고 timeout 트리거가 발생하면 이를 저장하고 처리한다.

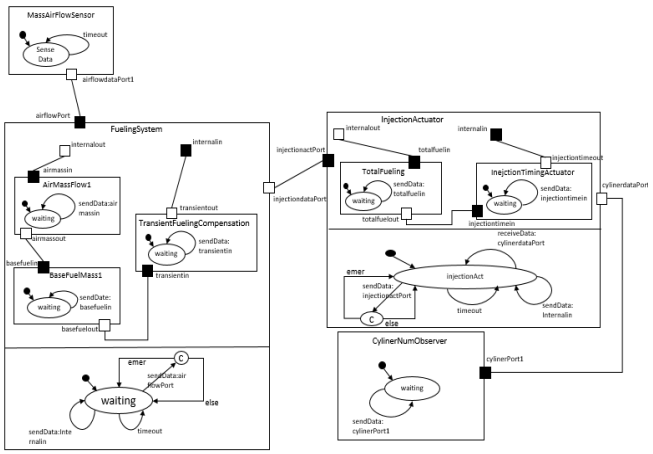


그림 5. ECU 시스템 중 FuelingSystem에 관련된 액터, 포트, 구조 모델 다이어그램

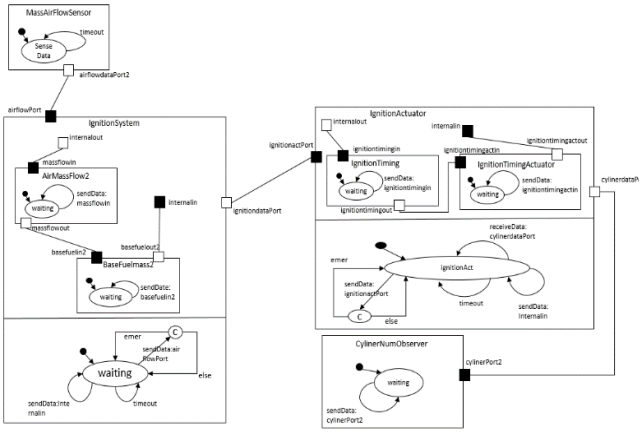


그림 6. ECU 시스템 중 IgnitionSystem에 관련된 액터, 포트, 구조 모델 다이어그램

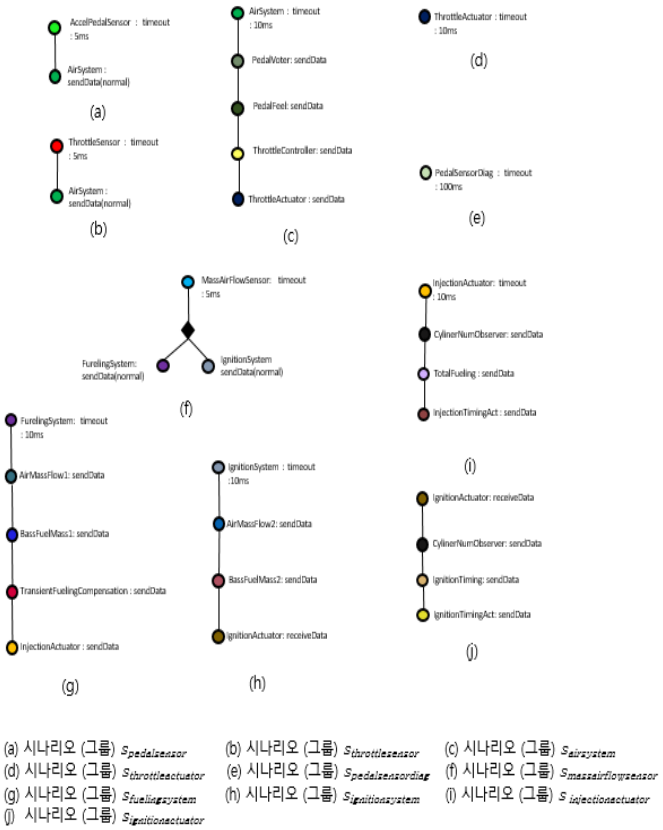
그림 5에서 MassAirFlowSensor는 공기의 흐름을 측정하여 FuelingSystem과 IgnitionSystem에서 연료를 주입할 양과 점화각도를 계산하는데 필요한 데이터를 생성한다. 생성한 데이터를 timeout트리거가 발생하면 외부 CRP로 메시지를 보내 FuelingSystem과 IgnitionSystem의 외부 RP로 전달한다.

FuelingSystem은 외부 RP, 내부 RP를 통해 메시지를 전달받아 저장한다. 또한 내부 CRP와 외부 CRP를 사용하여 메시지를 전달한다. 또한 FuelingSystem은 AirMassFlow1, BaseFuelMass1, TransientFuelingCompensation을 포함한 SubSystem을 포함하고 있다. Timeout 트리거가 발생되면 내부 CRP를 통해 SubSystem으로 메시지를 전달하고 SubSystem은 이를 처리하여 메시지를 FuelingSystem의 내부 RP로 전송한다. 그리고 FuelingSystem은 전달받은 메시지를 저장하고 이를 외부 CRP를 통해 InjectionActuator의 외부 RP에 전달한다.

InjectionActuator는 외부 RP, 내부 RP를 통해 메시지를 전달받아 저장한다. 또한 내부 CRP, 외부 CRP를 통해 메시지를 전달한다. 또한 InjectionActuator는 TotalFueling과 InjectionTimingActuator를 포함한 SubSystem을 포함하고 있다. Timeout 트리거가 발생되면 외부 CRP를 통해 CylinerNumObserver에 메시지를 전달하고 응답을 받는다. 그 후 내부 CRP를 통해 SubSystem으로 메세지를 전달하고 SubSystem은 이를 처리하여 메시지를 InjectionActuator의 내부 RP로 전송한다. 그리고 InjectionActuator은 메시지를 저장하고 처리한다.

그림 6의 IgnitionSystem은 외부 RP, 내부 RP를 통해 메시지를 전달받아 저장한다. 또한 내부 CRP와 외부 CRP를 사용하여 메시지를 전달한다. 또한 IgnitionSystem은 AirMassFlow2, BaseFuelMass2를 포함한 SubSystem을 포함하고 있다. Timeout 트리거가 발생되면 내부 CRP를 통해 SubSystem으로 메시지를 전달하고 SubSystem은 이를 처리하여 메시지를 IgnitionSystem의 내부 RP로 전송한다. 그리고 IgnitionSystem은 전달받은 메시지를 저장하고 이를 외부 CRP를 통해 IgnitionActuator의 외부 RP에 전달한다.

IgnitionActuator는 외부 RP, 내부 RP를 통해 메시지를 전달받아 저장한다. 또한 내부 CRP, 외부 CRP를 통해 메시지를 전달한다. 또한 IgnitionActuator는 IgnitionTiming과 IgnitionTimingActuator를 포함한 SubSystem을 포함하고 있다. Timeout 트리거가 발생되면 외부 CRP를 통해 CylinerNumObserver에 메시지를 전달하고 응답을 받는다. 그 후 내부 CRP를 통해 SubSystem으로 메세지를 전달하고 SubSystem은 이를 처리하여 메시지를 IgnitionActuator의 내부RP로 전송한다.



(a) 시나리오 (그룹) $S_{pedalsensor}$ (b) 시나리오 (그룹) $S_{throttlesensor}$ (c) 시나리오 (그룹) $S_{airsystem}$
(d) 시나리오 (그룹) $S_{throttleactuator}$ (e) 시나리오 (그룹) $S_{pedalsensordiag}$ (f) 시나리오 (그룹) $S_{massairflowsensor}$
(g) 시나리오 (그룹) $S_{fuelingsystem}$ (h) 시나리오 (그룹) $S_{ignitionsystem}$ (i) 시나리오 (그룹) $S_{injectionactuator}$
(j) 시나리오 (그룹) $S_{ignitionactuator}$

그림 7. eTrice 기반 ECU 모델의 시나리오 각 원은 주기를 가질 경우 “액터:메시지:주기“, 주기를 가지지 않을 경우 “액터:메시지”를 나타낸다.

그리고 IgnitionActuator은 메시지를 저장하고 처리한다.

결과적으로 ECU모델에서 4가지로 나누었던 AirSystem, PedalSensorDiag, FuelingSystem, IgnitionSystem 관련 시스템은 각각 7개, 1개, 9개, 8개의 액터로 구성되어 있다. 그러나 FuelingSystem 관련 시스템과 IgnitionSystem 관련 시스템에서 MassAirSensor, CylinerNumObserver 2가지 액터가 각각 1번씩 중복되었으므로, 총 23개의 액터로 구성되어 있다.

ECU모델에서는 AccelPedalSensor, ThrottleSensor, MassAirFlowSensor, PedalSensorDiag, AirSystem, FuelingSystem, IgnitionSystem, ThrottleActuator, InjectionActuator, IgnitionActuator 총 10개의 주기를 가지고 있는 액터가 있다. 이를 제외한 PedalVoter, PedalFeel, ThrottleController, AirMassFlow1, BaseFuelMass1, TransientFuelingCompensation, AirMassFlow2, BaseFuelMass2, CylinerNumObserver, TotalFueling, InjectionTimingAct, IgnitionTiming, IgnitionTimingact 총 13개의 액터들은 주기를 가지지 않는다. 그림 7에서 볼 수 있듯이 시나리오 그룹을 총 10개로 나누었다. 주기를 가진 액터를 기준으로 나누었으며 각 시나리오의 첫번째 액터들은 주기를 가지고 있는 액터들로 구성되어 있는 것을 볼 수 있다.

주기적인 상태천이를 가지고 있지 않은 액터들도 각기 다른 태스크로 매핑될 수 있지만 [15],[16]의 내용에 따라 주기성을 띄고 있는 액터를 기준으로 태스크를 매핑하고자 한다. 표 1과 같이 ECU 시스템 모델을 구성하고 있는 총 23개의 액터 중에서 주기를 가지고 있는 액터를 기준으로 시나리오에 따라 액터들을 태스크에 매핑하였다. CylinerNumObserver의 경우 주기를 띄지 않고 여러 시나리오에 사용된다. 때문에 CylinerNumObserver가 사용되는 시나리오 따라 매핑한 태스크들 중 WCET이 작은 태스크에 이 액터를 매핑하였다.

[14]에서 볼 수 있듯이 엔진이 최대 속도로 작동 할 경우 태스크의 종료시한을 만족시킬수 없으며, ThrottleActuator, InjectionActuator, IgnitionActuator들과 같은 구동자의 경우 시간에 매우 민감하기 때문에 최소한 코어를 3개 사용하여 각 구동자들과 관련된 태스크 8, 9, 10을 서로 다른 코어에 수동 매핑했다.

표 1. 객체 기반 멀티 태스킹에서의 태스크 (시간 단위:ms)

태스크/시나리오 번호	태스크에 매핑된 객체들	시나리오 주기/종료시한 $T_i = D_i$	시나리오 최악수행시간 C_i	시나리오 우선순위
1	PedalSensor	5	0.15	8
2	ThrottleSensor	5	0.06	9
3	MassAirFlowSensor	5	0.05	10
4	PedalSensorDiag	100	0.04	1
5	AirSystem, PedalVoter, PedalFeel, ThrottleController	10	0.99	2
6	FuelingSystem, AirMassFlow1, BaseFuelMass1, TransientFuelCompensation	10	1.85	3
7	IgnitionSystem, AirMassFlow2, BasFuelMass2	10	0.88	4
8	ThrottleActuator	10	0.89	5
9	InjectionActuator, TotalFueling, InjectionTimeActuator, CylinerNumObserver	10	0.87	6
10	IgnitionActuator, IgnitionTiming, IgnitionTimingActuator	10	1.59	7

4. 능동 객체 기반 멀티 태스킹에서의 우선순위 전파

우선순위 전파를 사용하지 않을 경우 시나리오의 우선순위가 지켜지지 않으면서 문제가 발생되게 된다. 이를 설명하기 위해 그림 8을 통해 설명한다. 그림 8에서 시나리오 번호는 표1을 참조하며, 시나리오 3을 통하여 우선순위 전파에 따른 예시를 보인다. 시나리오 3의 경우 태스크 3, 7, 6을 지나는 시나리오다. 그리고 표 2에서 볼 수 있듯이 태스크 3 우선순위 값은 시나리오의 우선순위와 같은 10이며, 태스크 7, 6 우선순위 값은 각각 4, 3 이다. 오른쪽 그림과 같이 시나리오 3의 우선순위를 전파하지 않는다면 태스크 7의 우선순위인 4를 이용하여 수행이 되면서 우선순위가 높은 시나리오 3의 응답시간이 길어지게 된다. 하지만 왼쪽 그림과 같이 우

그림 8. 우선순위 전파에 따른 시나리오 3 예시

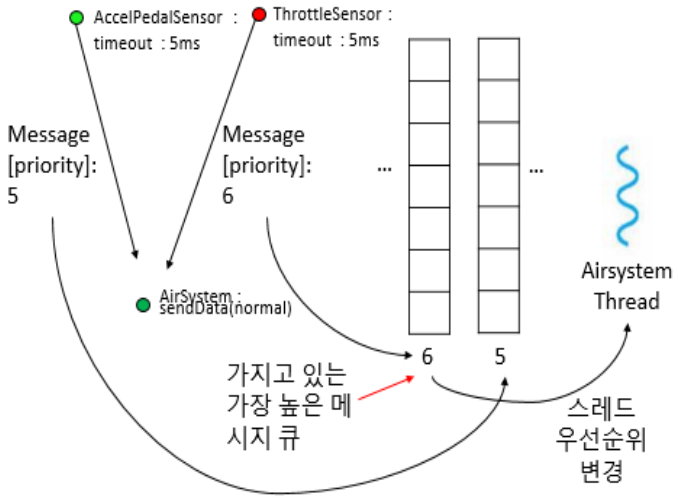


그림 9. 메시지의 우선순위 내용을 통한 우선순위 전달

선순위 전파를 사용하게 되면 시나리오 3의 우선순위를 보장하여 빠른 응답시간을 보이게 되므로 우선순위 전파가 필요하다.

본 논문에서는 시나리오를 기반으로 한다. 그렇기 때문에 시나리오 단위로 우선순위를 갖는다. 시나리오와 관련된 태스크의 경우 시나리오가 진행될 때 시나리오의 우선순위를 전파받아 수행될 수 있어야한다. 그러나 eTrice에서는 우선순위 전파 기능이 없었기 때문에 이를 구현하였다. 또한 시나리오 기반으로 우선순위를 전파할 경우 여러 시나리오에서 쓰이는 태스크가 생기게 된다. 그래서 여러 우선순위 전파에 따른 다중 레벨 큐를 구현하였다. 또한 우선순위는 Posix Pthread 라이브러리가 지원하는 0~99까지의 범위로 하였다.

그림 9와 같이 포트를 통하여 메시지를 전달할 때 시

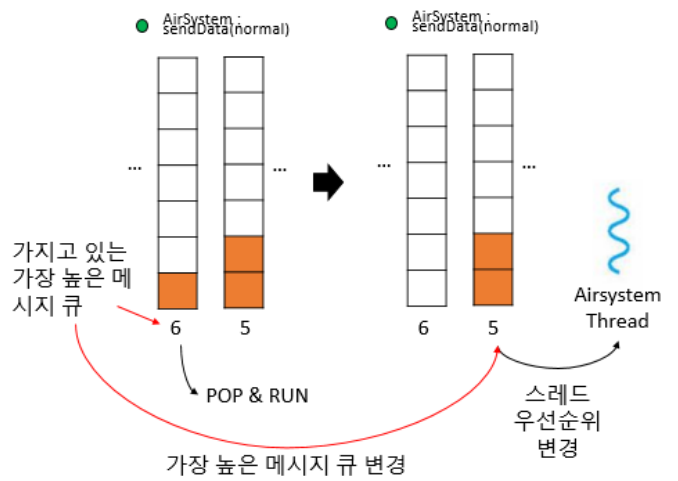


그림 10. 다중 레벨 큐에서 태스크가 수행할 메시지

나리오의 우선순위를 담아 전달했다. 그래서 메시지 내용 중 우선순위와 관련된 내용과 전달받는 스레드의 우선순위와 비교하여 더 높은 우선순위로 스레드의 우선순위를 높인다. 이와 같이 전달받은 메시지의 우선순위를 통하여 스레드의 우선순위를 변경하고, 우선순위를 전달받는 Task가 여러 Task로부터 우선순위를 전달받는 경우 한 개의 메세지 큐로는 이를 수행하기 어려워지기 때문에 다중 레벨 큐를 0~99까지 우선순위 별로 만들어 놓고 메시지의 우선순위에 따라 알맞은 레벨 큐에 메시지를 넣었다.

그림 10과 같이 코어에서 스레드가 수행되면 스레드는 자신의 다중 레벨 큐들 중 메세지를 가지고 있는 큐들을 탐색한다. 그리고 탐색된 큐들 중 가장 높은 우선순위를 가지고 있는 큐에서 메시지를 꺼내어 수행하게 된다. 또한 이 메시지가 가장 높은 우선순위 큐에서 마지막 메시지라면 이를 꺼낸 후 다음 높은 우선순위 큐의 우선순위로 스레드의 우선순위를 낮추게 된다.

5. 능동 객체 기반 태스크의 멀티 코어 매핑

```
1  foreach (s_i ∈ S) {
2      foreach (p_j ∈ P_i) {
3          STask={}
4          foreach (τ_j ∈ L_j) {
5              if(τ_j in Sτ_i ) {
6                  R_{i,j} = R_{i,j} + C_{τ_j}
7                  R_{i,j} = R_{i,j} + C*_{τ_j}
8                  STask <- τ_j
9              }
10         } //end-foreach
11         foreach (τ_j ∈ L_j) {
12             foreach (τ_k ∈ STask ) {
13                 if(π_k < π_j)    R_{i,j} = R_{i,j} + C_{τ_j}
14             }//end-foreach
15         } //end-foreach
16         R_i = R_i + R_{i,j}
17     } //end-foreach
18 } //end-foreach
```

그림 11. 최악응답시간 계산 알고리즘 : 알고리즘의 표기법은 2장의 표기법과 별개이며 다음과 같다. $\boldsymbol{S}$: 시나리오 집합, $\boldsymbol{S\tau}$: 시나리오가 수행하는 태스크 집합, $\boldsymbol{P}$: 시나리오가 수행하는 태스크가 매핑된 코어집합, $\boldsymbol{\tau}$: 태스크, $\boldsymbol{L}$: 코어에 매핑된 태스크 집합, $\boldsymbol{C_\tau}$: 태스크의 수행시간, $\boldsymbol{C^*_\tau}$: 태스크의 크리티컬 섹션, $\boldsymbol{\pi}$: 우선순위, $\boldsymbol{R}$:응답시간

2장에서 설명하였듯이, 각 시스템의 응답시간은 시나리오의 수행으로 이루어 지며, 시나리오의 종료 시한이 시스템의 시간 제약 조건으로 연결된다. 3장의 표 1과 같이 10개의 태스크가 존재하고 이 태스크들을 표 2와 같이 멀티 코어로 매핑하였다. 또한 멀티 코어에서 GS-WF 알고리즘을 사용하였고, 이 GS-WF 매핑기법을 사용하여 wait-free에 의해 전역자원 블로킹 시간을 무시하고, 이 방법의 기준인 최대의 슬랙을 도출하는 기준으로 3개의 코어로 자동할당 시켰다. 하지만 자동 할당하기에 앞서 구동자들은 다른 코어에 매핑할 필요성이 있으므로, ThrottleActuator, InjectionActuator, IgnitionActuator에 대해 각각 3개의 다른 코어로 먼저 매핑하였다. 그리고 GS-WF 매핑 기법을 시나리오 단위 우선순위를 전파의 여부에 따라 나누어 시나리오 단위 우선순위 전파가 성능에 미치는 영향에 대해 비교하였다. 시나리오 단위 우선순위 전파는 시나리오의 우선순위를 시나리오를 진행하며 사용하는 태스크에 전달하여 사용하는 것을 말한다.

ECU시스템을 구성하는 10개의 시나리오에 대한 최악 응답시간을 그림 11과 같이 이론적으로 분석하였다. 시나리오와 연관된 모든 코어 집합 P 를 탐색하여 시나리오의 최악응답시간을 구한다. 시나리오와 연관된 코어에서 시나리오와 연관된 태스크$\boldsymbol{\tau}$를 찾아 수행시간을 최악 응답시간에 더 하고 $\boldsymbol{\tau}$의 가장 큰 임계구역을 찾아 블로킹 타임으로써 최악응답시간에 더한다. 그리고 그 $\boldsymbol{\tau}$들을 Stask에 따로 저장하여 해당 코어에서 시나리오와 연관된 $\boldsymbol{\tau}$들보다 큰 우선순위를 가지고 있는 $\boldsymbol{\tau}$들의 수행시간을 더하여 간섭시간을 구한 후 이를 최악응답시간에 더한다. 이와 같이 분석한 결과를 표 2 분석에 제시하였다.

분석한 결과를 토대로 슬랙 산술 평균 계산하여 성능을 평가해 보았을 때 GS-WF 알고리즘을 적용한 멀티 코어에서 시나리오 단위 우선순위를 전파시킨 기법이 우선순위를 전파를 시키지 않은 기법보다 슬랙 산술 평균의 값이 더 컸다. 결과적으로 시나리오에 따른 우선순위를 전파를 사용하는 알고리즘이 더 좋은 성능을 낸 것을 보여준다.

6. eTrice 구현 실험 결과

표 2 의 매핑 방법을 기반으로 태스크들을 표 2 실험과 같이 실제 eTrice를 사용한 실시간 객체 지향 모델에 적용시켜 실험을 수행하였다. 실험 환경은 Linux v. 4.13.0-36-generic, eTrice v. 1.1.2, Intel i7-4770k CPU 3.50GHz (8 core), DDR3 4GB RAM이며, eTrice 언어는 C를 사용하였다. 실험 결과, 이론적으로 멀티 코어에서 GS-WF 알고리즘을 사용하고 우선순위 전파를 사용하지 않으면 슬랙 산술 평균이 0.70 이 되고 우선순위 전파를 사용한다면 슬랙 산술 평균이 0.76 이 되어 성능이 향상될 것으로 예상하였다. 또한 모든 태스크가 종료시한을 만족시킬 것으로 예상하였다. 이를 토대로 실제 eTrice에 구현된 ECU모델 태스크들을 표 2 와 같이 매핑하였을때 우선순위의 전파여부와 관계없이 모두 종료시한은 만족시키지만 이론적 분석에

표 2. 다양한 코어에 할당된 태스크들 및 시나리오 응답시간 분석(시간 단위:ms)

태스크 번호/ 시나리오 번호	시나리오 우선순위	멀티코어_GS-WF_ 우선순위 전파X					멀티코어_GS-WF_ 우선순위 전파O				
		매핑코어	응답시간		슬랙산술평균		매핑코어	응답시간		슬랙산술평균	
			분석	실험	분석	실험		분석	실험	분석	실험
1	8	3	1.36	1.57	0.70	0.69	3	0.77	0.49	0.76	0.75
2	9	3	1.20	1.57			3	0.61	0.23		
3	10	1	2.97	3.75			3	1.19	1.38		
4	1	3	1.83	0.07			3	1.87	0.08		
5	2	1	3.88	1.87			1	3.84	2.03		
6	3	2	4.70	4.29			2	4.70	5.26		
7	4	3	3.78	4.07			3	3.81	2.46		
8	5	1	1.22	1.59			1	1.18	1.70		
9	6	2	1.90	1.90			2	1.90	4.22		
10	7	3	3.01	3.47			3	3.04	4.20		

비해 최악응답시간이 긴 태스크가 있었다. 또한 슬랙 산술 평균 또한 모두 이론적 분석보다 0.01 정도 낮은 결과를 확인할 수 있었다.

실험을 Linux 운영체제에서 진행하면서 우선순위 전파를 적용한 경우와 적용하지 않은 경우 모두 모든 스레드가 우선순위 메시지 큐를 가지고 있고, 이 메시지 큐에 대해 enqueue/dequeue를 수행할 때 동기화를 위해 뮤텍스를 사용하였다. 그러나 이론적 분석의 GS-WF는 이 메시지 큐를 wait-free로 동기화를 하여 뮤텍스를 사용하지 않았기 때문에 이론적 부하가 작을 수 밖에 없었다. 또한 우선순위를 전파하기 위해 사용하는 pthread_setschedparam()이 futex(fast userspace mutex)를 사용하면서 뮤텍스를 추가적으로 사용하였다. 이로 인해 본 논문의 시스템 가정에서 무시하였던 뮤텍스에 의한 문맥전환 부하가 많아지면서 응답시간이 길어진 것으로 보인다.

실험적 부하가 큰 문제를 해결하기 위한 향후 방안으로 액터 기반 멀티 코어 멀티 태스킹에서 스레드 별 메시지 큐를 뮤텍스가 아닌 non-blocking 동기화를 구현하거나 MSRP(Multiprocessor Stack Resource Policy)[17]을 구현하여 사용하는 방법을 생각해 볼 수 있다. 또한 액터 기반이 아닌 시나리오 기반 멀티 코어 멀티 태스킹으로 수행하여 메시지 큐의 뮤텍스 부하를 줄일 수 있다.

실험적 결과에서 시나리오 단위의 우선순위를 전파한 경우와 우선순위를 전파하지 않은 경우를 비교해 보았을 때 우선순위를 전파하지 않은 경우 슬랙 산술 평균이 0.69, 우선순위를 전파한 경우 슬랙 산술 평균이 0.75 로 측정되었다. 결과적으로 우선순위를 전파하였을 경우 슬랙 산술 평균이 이론적 분석과 마찬가지로 0.06 향상되어 이론적 분석의 추세와 실험적 분석의 결과가 일치하는 것을 확인하였다.

7. 결론

본 논문에서는 오픈 소스 실시간 객체지향 모델링 도구인 eTrice를 이용한 자동차 ECU 시스템을 객체기반 멀티 코어 시스템 사례연구로서 제시하였다. 이 시스템은 3 개의 구동자를 각각 다른 코어에 매핑하여야하는 시스템이기 때문에 싱글 코어로는 불가능한 사례이다.

[18]의 향후 연구로서 멀티 코어 시스템에서 시나리오 단위의 우선순위 전파를 하였을 경우와 우선순위 전파를 하지 않았을 경우에 대해 성능 비교 측정을 수행하였고, 이를 통해 메시지를 이용하여 시나리오 단위의 우선순위를 태스크로 전달할 경우 간섭 시간은 감소시킬 수 있음을 보였다.

그러나 사례 연구에서 객체 기반 멀티 코어 매핑을 하면서 액터간에 메시지를 전달하는 상태천이와 우선순위 전달 방법에서 뮤텍스를 많이 사용했다. 이로 인해 뮤텍스 문맥교환 부하가 많이 증가했고, 멀티 코어 시스템에서 뮤텍스 부하 때문에 성능이 나빠질 수 있음을 보였다.

향후 연구로서, 많은 뮤텍스를 사용하게 되면서 발생하는 문맥전환 부하를 줄이기 위해 eTrice 실시간 객체 지향 모델링 도구로 ECU 시스템을 액터 기반이 아닌 시나리오 기반 멀티 태스킹을 이용하여 멀티 코어 매핑을 하는 것에 대한 연구를 진행하고 있다.

참고문헌

[1]https://www.ibm.com/support/knowledgecenter/en/SSSHUF_8.0.0/welcome.html

[2]https://www.ibm.com/support/knowledgecenter/ko/SSB2MU_8.1.5/com.ibm.rhp.overview.doc/topics/rhp_c_po_rr_product_overview.html

[3] https://www.eclipse.org/etrice/

[4] https://www.protos.de/

[5] S. Kim, “Synthesizing Multithreaded Code from Real-Time Object-Oriented Models via Schedulability-Aware Thread Derivation,” IEEE Transactions on Software Engineering, Vol. 40, No. 4, pp. 413-426, April 2014.

[6]https://en.wikipedia.org/wiki/Real-Time_Object-Oriented_Modeling

[7] Karthik Lakshmanan, Ragunathan Rajkumar, John P. Lehoczky, “Partitioned Fixed-Priority Preemptive Scheduling for Multi-Core Processors”, 2009 21st Euromicro Conference on Real-Time Systems, 2009

[8] Institute for Electrical and Electronic Engineers, IEEE Std. 1003.1c-1995 POSIX Part 1: System application program interface - amendment 2: threads extension, 1995.

[9] Z. Al-bayati, Y. Sun, H. Zeng, M. Di Natale, Q. Zhu and B. Meyer, "Task placement and selection of data consistency mechanisms for real-time multicore applications," IEEE Real-Time and Embedded

Technology and Applications Symposium, pp. 172-181, 2015.

[10] 김세화 “객체 기반 멀티 태스킹에서의 실시간 객체 지향 모델의 멀티 코어 매핑: 축구 로봇 시스템을 이용한 사례 연구”, 정보산업공학논문집 제 30 집, 2017.8

[11] 김세화 “시나리오 기반 멀티 코어 멀티 태스킹에서의 실시간 객체 지향 모델의 최악 응답 시간 분석”, 정보산업공학논문집 제 31 집, 2018.2

[12] Piotr Dziurzanski, “Benchmarking, System Design and Case-studies for Multi-core based Embedded Automotive Systems”, cs.DC,2016

[13] Patrick Frey, “Case Study: Engine Control Applicataion” , Ulmer Informatik-Berichte, 2010

[14] Mikael A° sberg, “Towards Hierarchical Scheduling in AUTOSAR”, etfa, 2009

[15] M. Saksena, P. Freeman, and P. Rodziewicz, "Guidelines for automated implementation of executable object-oriented models for real-time embedded control systems," Proceedings of IEEE Real-Time Systems Symposium, pp. 240-251, 1997

[16] D. Gaudrean and P. Freedman, "Temporal analysis and object-oriented real-time software development: A case study with ROOM/objectime," Proceedings of IEEE Real-Time Systems Symposium, pp. 110-118, 1996.

[17] P. Gai, G. Lipari, and M. D. Natale. “Minimizing memory utilizationof real-time task sets in single and multi-processor systems-on-a-chip.”In Proc. 22nd IEEE Real-Time Systems Symposium, 2001.

[18] 김경률, 김세화, “오픈 소스 실시간 객체 지향 모델링 도구에서의 객체 기반 멀티 코어 매핑 사례 연구”, 정보산업공학연구소, 2018

패혈증 진단용 REBA 검사 자동화를 위한 영상인식 모바일 어플리케이션 개발

류원철 ○ 황상원 서강원 남영광

연세대학교

hkjkkl@naver.com, arsenal@yonsei.ac.kr, seogs12@naver.com, yknam@yonsei.ac.kr

Development of mobile application for automated image recognition of sepsis REBA diagnosis

Won-Cheol Ryu○ Sang-Won Hwang Gang-Won Seo Young-Gwang Nam

Yonsei University

요 약

본 논문에서는 패혈증 진단용 REBA 검사 키트의 결과를 모바일 카메라로 촬영하여 원인균 검출 결과를 출력하는 모바일 어플리케이션 개발을 목표로 한다. 모바일 상에서 이미지 처리를 수행하기 위해 Android와 OpenCV 라이브러리를 사용하여 어플리케이션을 개발하였다. 모바일 디바이스를 통해 촬영한 이미지는 카메라의 상태와 주위 환경에 따라 이미지에서 색이 흐릿하게 나타나므로, Histogram Stretching 알고리즘을 적용하여 색의 대비를 강화하였다. 색의 대비가 강화된 이미지를 대상으로 색의 발현을 인식하며, Edge Detection 방법을 적용하여 이미지에서 색의 영역을 검출하였다. REBA 검사 결과에서 사각형의 형태로 색이 발현되므로, Edge Detection 방법 중 Prewitt Mask를 사용하여 색의 경계를 검출하였다. 이미지의 노이즈를 제거하기 위한 알고리즘도 적용하여 인식률을 향상시켰다.

1. 서 론

패혈증(Sepsis)은 세균 등의 감염균이 체내로 들어가 번식하면서 생산하는 독소에 의해 나타나는 중독 중세나, 세균이 혈액의 순환에 의해 전신으로 퍼져 2차적으로 여러 장기에 감염을 일으킨 전신 감염 상태를 말한다.

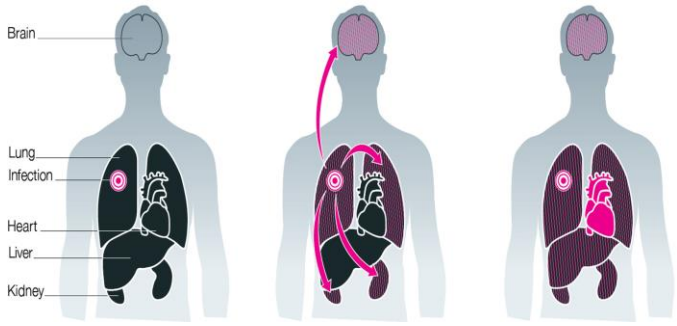


그림 1. 전신감염반응증후군(1단계), 중증패혈증(2단계), 패혈성 쇼크(3단계)의 양상

(그림 1)은 패혈증의 양상을 단계적으로 나타낸 것이다. 1단계, 세균 감염에 의한 국소감염(예, 폐에서)은 인체의 국소 방어기전에 의해 해결되나, 병원체와 생성된 독소는 본래의 감영 부위에서 벗어나 순환기로 침입하여 전신 감염 반응 증후군(SIRS, Systemic Inflammatory Response Syndrome)을 유발한다. 2단계는 중증패혈증으로 개별 장기의 파손 및 손상이 시작된다. 마지막 3단계에서는 여러 장기의 연속/동시 기능정지, 심혈관계 부전으로 인해 급성 혈압저하인 패혈성 쇼크(Septic Shock)가 발생한다.

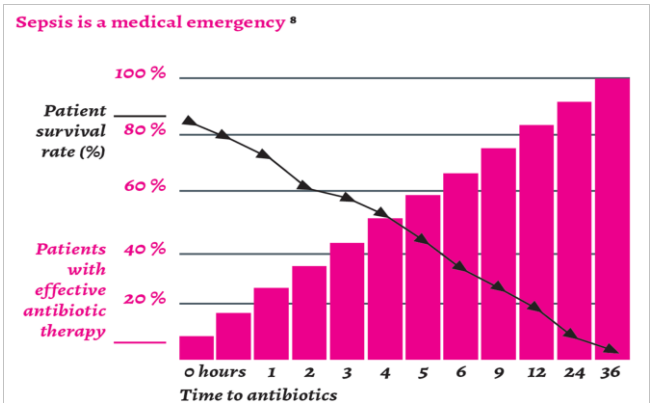


그림 2. 항생제투여 시간에 따른 환자 생존률

패혈증은 특정 병원체에 의한 질병이 아니며, 그람음성세균과 그람양성세균 및 진균류등의 감염에 대한 일련의 연속적 염증면역체계 반응으로 촉발되는 반응이다. (그림 2)의 그래프를 보면 원인균에 대한

항생제의 투여가 1시간씩 지연될 때마다 환자의 생존율이 7.6%씩 감소한다. 패혈증의 원인균의 신속하고 정확한 동정과 효과적인 항생제의 사용이 패혈증 환자의 생존율에 크게 영향을 끼치는 것을 알 수 있다.

기존의 패혈증 진단 방법은 혈액배양검사 및 고체배지검사 그리고 항생제 감수성검사를 수행하는데, 검사결과가 나오기까지 최소 2일에서 7일 정도 시간이 소요되며 여러 번의 검사가 필요하여 검사비용 또한 적지 않다.

이와 같은 문제를 해결하기 위해 다중 바이오마커를 이용한 분자생물학적 검사법인 역교잡반응(Reverse blot hybridization assay, 이하 REBA)을 이용하여 신속하고 정확하게 패혈증 원인균을 동정하고 동시에 항생제 내성 여부를 판별할 수 있는 패혈증 진단용 REBA 검사가 개발되었다[1, 2, 3, 4, 5].

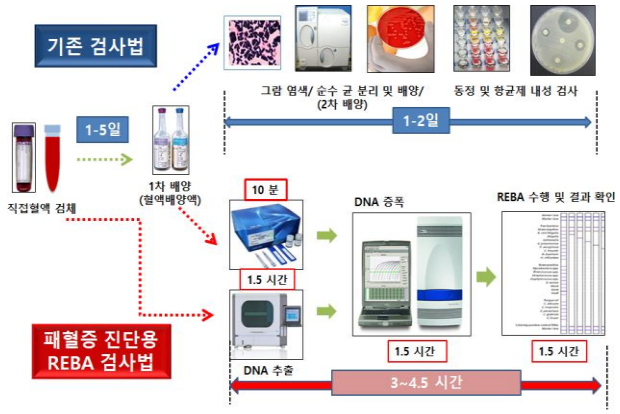


그림 3. 기존 검사법과 REBA 검사법의 과정 및 소요시간 비교

위의 (그림 3)은 패혈증 진단에 사용하는 기존 검사법과 REBA 검사법의 과정 및 소요시간을 비교한 그림이다. 기존 검사법은 2~7일 정도의 시간이 필요한데 반해, REBA 검사는 3~4.5시간 내에 검사 결과가 나오는 것을 알 수 있다. REBA 검사법은 검사시간이 상당히 단축되지만, REBA 검사 수행 후에 검사 결과를 별도의 용지에 부착하여 육안으로 직접 판독해야 한다. 용지에 부착할 때 노이즈가 생기거나, 오랜 시간 보관할 경우에 색의 선명도가 감소하게 된다. 이와 같은 요인에 의해 판독하는 개인에 따라 판독결과가 달라지는 문제점이 있다. 본 논문에서는 REBA 검사 결과를 모바일 디바이스를 통해 촬영하여 원인균을 판독하는 어플리케이션을 개발하였다.

아래의 (그림 4)는 어플리케이션의 처리 과정을 나타낸 것이다. 모바일 디바이스를 통해 REBA 검사 결과를 촬영한 이미지에서 REBA 영역을 추출하고, 추출한 이미지에 대하여 색의 대비를 강화한다. 색이 강화된 이미지에서 색의 경계를 인식하여 색의 영역 후보군을 생성하고, 그 중 노이즈로 판단되는 영역을 제거한다. 최종적으로 판단된 색의 영역에 대하여 패턴열로 변환하여, 원인균의 색 발현 패턴과 비교하여 검출된 원인균을 판단한다. 판독 결과는 어플리케이션의 화면으로 출력됨과 동시에 서버의 Database로 저장되며, 판독 REBA 이미지 파일과 판독 이미지와 판독결과를 포함하고 있는 PDF 파일은 서버의 저장소로 저장된다.

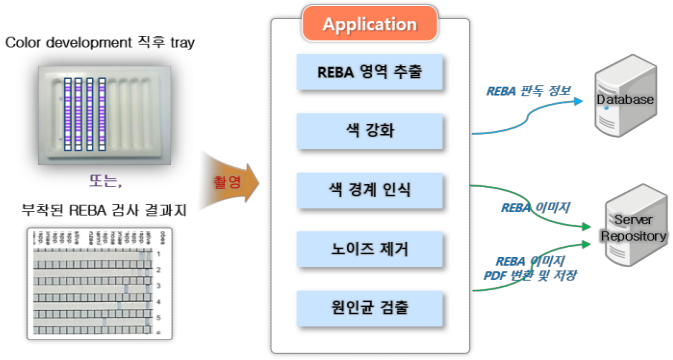


그림 4. 어플리케이션의 처리 과정

2장에서는 본 연구와 관련 있는 연구에 대하여 설명한다. 3장에서는 본 논문에서 개발한 시스템에 대해 설명한다. 4장은 본 논문에서 개발한 시스템과 기존에 연구된 방법과의 비교를 수행한다. 비교를 통해서 시스템의 성능을 평가하였다. 5장에서는 결론 및 향후 진행할 연구에 대하여 논하였다. 6장에서는 본 연구를 진행하면서 참고한 문헌들에 대해 서술하였다.

2. 관련 연구

이미지에서 색을 인식하는 기술들은 영상처리(Image Processing) 분야에서 활발히 연구된 분야이며, 색의 경계면을 인식하여 색 영역을 검출하기 위해서 윤곽선 검출(Edge Detection) 방법[6] 외에도 다양한 방법을 적용한 연구결과들이 많이 발표되었다.

이미지를 벡터로 변환하여 벡터의 방향변화를 계산하여 윤곽선을 판단하는 방법[7], 이미지에서 관심있는 색 영역에 대하여 분류하여 원하는 색 영역을 추출하는 방법[8], 벡터의 차와 morphological gradient operator를 이용해 이미지에서 윤곽선을 추출하는 방법[9] 등이 제안되었다.

Dai 등은 픽셀의 로컬 엔트로피를 계산하여 엔트로피가 높게 나타나는 픽셀을 Edge 픽셀로 판단하여 색의 경계를 인식하는 방법을 제안하였다[10]. 이미지의 각 픽셀과 주변의 8개의 픽셀, 3x3 윈도우에 대하여 로컬 엔트로피를 계산한다. 로컬 엔트로피가 낮을수록 해당 윈도우의 값의 분산(dispersion)이 커진다는 이론에 적용한다. 윤곽선 픽셀이 윈도우에 포함되면 로컬 엔트로피가 높아지므로 로컬 엔트로피가 높은 픽셀을 윤곽선 픽셀로 판단하여 이미지에서 색의

경계를 추출한다. 주변과 색의 차이가 크지 않은 부분에 대해서는 경계 검출이 어려운 단점이 있다.

Levin 등은 매팅 기법을 적용하여 배경 이미지와 물체 이미지를 구분하는 방법을 발표하였다[11]. 이미지상에 배경과 물체 간의 구분선을 사용자가 입력해줘야 하는 한계가 있어 자동으로 색을 인식해야 하는 본 연구에는 적용하기 힘들다.

Chen 등은 Kuwahara Filter와 Sobel Operator를 적용한 색 경계 인식 방법을 제안하였다[12]. 이미지의 노이즈를 제거하기 위해 Kuwahara Filter를 적용해 이미지 스무딩(smoothing) 과정을 진행한다. 이미지를 RGB 각 채널로 나눈 후 Sobel Operator를 적용해 각 채널별로 윤곽선 맵을 생성한다. 각 채널별로 생성된 윤곽석 맵을 하나로 합성하고 이미지 임계처리 방법 중 하나인 적응 임계처리 알고리즘(Adaptive Thresholding Algorithm)과 Edge Thinning 방법을 적용하여 윤곽선 맵에서 확실한 윤곽선만을 추출하여 윤곽선 맵의 신뢰도를 높였다. PC 환경에서 개발이 진행되어 실시간 처리가 필요한 모바일 환경에서는 연산속도의 부족으로 적용이 어렵다.

본 논문에서는 Edge Detection을 통해 색의 발현을 인식하고, 인식된 색 영역 후보군에서 노이즈 영역을 판단한 후 제거하여 색의 발현을 인식하는데 있어 거짓양성 결과를 제거하는 방법을 사용하였다. 노이즈 영역을 판단하는 알고리즘은 REBA 검사 결과 이미지를 분석하여 분석 정보를 기반으로 개발하였다.

3. REBA 판독 시스템 구현

Java GUI를 이용하여 REBA 검사결과 이미지에서 색 인식 알고리즘을 개발하였다.

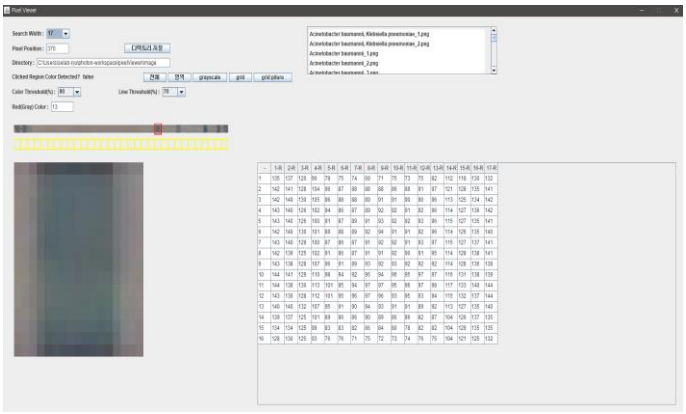


그림 5. Java GUI 프로그램 화면

위의 (그림 5)는 Eclipse를 통해 개발한 GUI 프로그램의 화면이다. GUI를 통해 이미지의 일부 영역에 대해서 이미지를 확대하고 영역의 Pixel값을 분석하여 알고리즘을 개발하였다.

안드로이드 모바일 어플리케이션을 개발하기 위해서 대표적인 개발도구인 Android Studio를 사용하였다. OpenCV 라이브러리를 통해 영상처리에 필요한 기능을 사용하였다. OpenCV 라이브러리는 C 언어 기반 라이브러리이므로 Android Studio에서 사용하기 위해 NDK(Native Development Kit)를 설치하고, JNI(Java Native Interface)를 구축하였다. NDK를 통해 안드로이드 환경에서 C언어 코드를 컴파일하고, JNI를 통해 Java 코드에서 C 코드를 호출하여 OpenCV 라이브러리를 사용하였다.

3.1 원인균 패턴 테이블 생성

REBA 검사 키트에서는 원인균을 판독하기 위해서 REBA 검사 용지를 32개의 영역으로 나눠서 색을 발현시킨다. REBA 검사 키트에는 양 끝 단에 시작과 끝, 방향을 나타내기 위한 각 2개의 영역이 있으며 중간부의 원인균 영역은 그람음성세균, 그람양성세균, 진균류 3개의 종류로 나뉜다. 개별 영역들의 색 발현 조합을 통해서 원인균을 판독한다(그림 6).

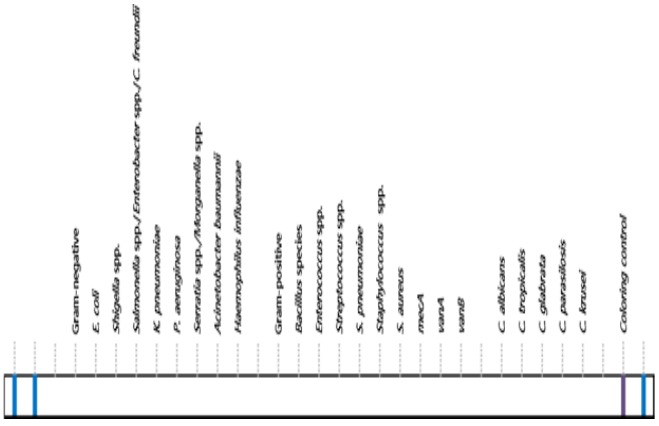


그림 6. REBA 검사 키트 각 영역별 원인균

이를 어플리케이션에 적용하기 위해서 원인균에 따라 결정되는 색 발현 조합들을 분석하였다. 색 발현 조합을 0과 1의 패턴으로 변환하여 파일로 저장하였다. 원인균 패턴 테이블은 원인균과 색 발현 패턴의 매칭으로 이루어져 있으며, 저장된 파일을 어플리케이션이 실행될 때마다 읽어 REBA 이미지 인식 후 원인균을 판독할 때 사용하였다. 아래의 (표 1)은 색의 발현이 패턴으로 변환된 예시이다.

표 1. 원인균 패턴 예시

원인균	패턴
Not Detected	11 0 000000000 0 000000000 0 00000 0 11
Gram Negative	11 0 100000000 0 000000000 0 00000 0 11

Escherichia Coli	11 0 110000000 0 000000000 0 00000 0 11
Shgella species	11 0 101000000 0 000000000 0 00000 0 11

색 인식 결과를 패턴으로 변환하여 인식함으로써 패혈증 검사 방법 외에도 병원균 검사 방법에 대해서 동일한 방법으로 적용할 수 있다. 검사 방법의 색 발현 영역을 분석하고, 병원균에 대하여 패턴 테이블을 형성하여 여러 종류의 검사 방법에 대하여 본 시스템을 적용할 수 있다.

3.2 어플리케이션 GUI 및 REBA 이미지 영역 추출

모바일 디바이스로 촬영한 이미지에서 REBA 이미지 영역을 추출하기 위해 아래의 (그림 7)과 같이 어플리케이션의 GUI에 노란색의 사각형 그리드를 그려 촬영할 영역을 표시하였다.

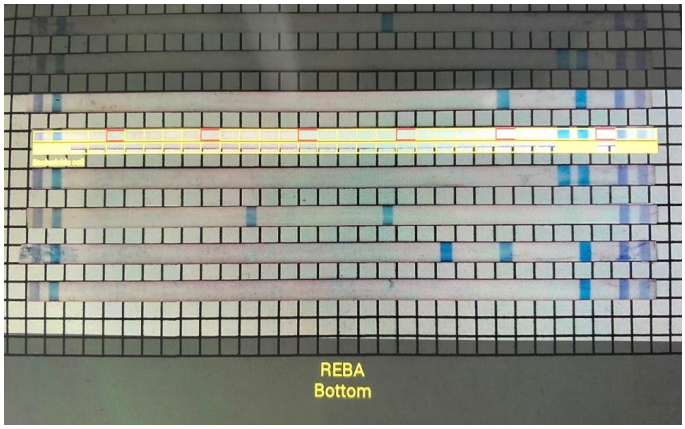


그림 7. 어플리케이션 GUI

GUI에 그려지는 영역은 REBA 이미지 영역, 패턴 출력 영역, 판독 원인균 출력 영역 3가지로 구분한다. REBA 검사는 32개의 색 발현을 조합하여 원인균을 판단하므로 REBA 이미지와 패턴 출력 영역은 32칸의 그리드로 표현하였다. REBA 이미지 영역에 맞춰서 REBA 검사 결과를 촬영하면 해당 영역의 이미지를 추출하여 분석하였다.

3.3 이미지 색 강화

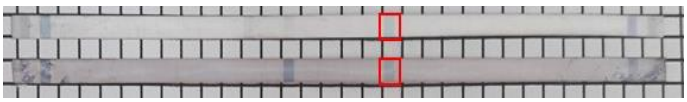


그림 8. REBA 검사 결과의 선명도 감소

REBA 검사 키트를 통한 검사 결과는 (그림 8)과 같이 시간이 지남에 따라 색이 발현된 부분의 선명도가 감소하는 문제점이 있다.

색의 대비를 강화하는 알고리즘을 선택하기 위해서 포토샵을 통해 이미지의 속성을 분석하였다. 분석한 결과 REBA 검사 결과 이미지들의 히스토그램 분포가 중간부에 집중되어 있는 것을 확인하였다. 중간부에 집중되어 있는 히스토그램 분포를 수정하기 위해서 Histogram Stretching 방법[13]을 사용하였다.

Histogram Stretching 방법은 특정 구역에 집중되어 있는 이미지의 히스토그램을 전체 영역에 분포되도록 조정하여 색과 밝기의 대비를 강화는 알고리즘이다. Histogram Stretching 알고리즘을 적용하기 위해 이미지를 grayscale 이미지로 변환하였다. 알고리즘의 단계는 아래와 같다.

① 입력된 grayscale 이미지에 대하여 픽셀 값이 가장 높은 값(high)과 가장 낮은 값(low)을 찾는다.

② high와 low값을 사용하여 이미지에 적용할 scale 값을 계산한다.

$$scale = 255 / (high - low)$$

③ 원본 이미지에 아래의 수식을 적용하여 색의 대비가 강화된 이미지를 얻는다.

$$pixel[i][j] = (pixel[i][j]-low) * scale$$

그림 9. Histogram Stretching 적용 결과
(a) 원본 이미지 (b) 적용 이미지

위의 (그림 9)은 Histogram Stretching을 REBA 이미지에 적용한 결과이다. 원본 REBA 이미지에서는 색의 발현이 선명하게 나타나지 않지만, 알고리즘 적용 후에는 색의 발현이 선명하게 나타난다.

3.4 이미지 색 인식

REBA 이미지에서 색의 발현은 위의 (그림 9)와 같이 사각형의 모양으로 나타난다. REBA 키트는 32개 영역에 색 발현이 나타나므로 REBA 이미지 너비를 32개의 영역(Window)로 나누어 각 영역에 대해서 색의 발현을 인식하였다. 색 발현을 판독하기 위해서 색의 경계(Edge)를 인식하여 색의 영역(Region)을 추출하는 방법을 택하였다. 인식된 Edge에 대해서 배경에서 색으로 변하는 경계(Open Edge) 색에서 배경으로

변하는 경계(Close Edge)로 분류하여 윈도우에 open, close Edge가 모두 존재하고, 일정 간격 이상의 거리가 존재할 경우 색 영역으로 판단하였다. 색의 경계는 Edge Detection 알고리즘을 적용하여 인식하였다.

이미지에서 Edge는 색의 강도가 급격하게 변하는 부분을 말하는데, 색의 강도 변화를 판단하기 위해서 이미지를 미분하여 값의 변화량 (기울기)을 계산하였다. 영상에서는 데이터가 일정 간격으로 나열되기 때문에 수학적인 미분 연산 대신 인접한 화소의 차이를 취하는 연산으로 대체하였다. Operator Mask를 통해 인접한 화소의 차이를 계산하였다. REBA 이미지는 색의 발현이 사각형의 형태로 나타내므로 수평과 수직 Edge에 대해 민감하게 반응하는 Prewitt Mask[6]를 사용하여 미분하였다. Prewitt Mask는 아래의 (그림 10)의 (a)와 같다.

$$G_x=\begin{bmatrix}-1 & 0 & +1\\ -1 & 0 & +1\\ -1 & 0 & +1\end{bmatrix},\quad G_y=\begin{bmatrix}+1 & +1 & +1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix}$$

(a)

$$|G|=\sqrt{G_x^2+G_y^2}$$

(b)

그림 10. (a) Prewitt Mask, (b) 변화량 계산 수식

Prewitt Mask를 사용하여 x,y 각 축에 대한 변화도를 계산한 후 해당 픽셀의 변화량을 얻기 위해 위의 (그림 10)의 (b) 수식을 적용하였다.

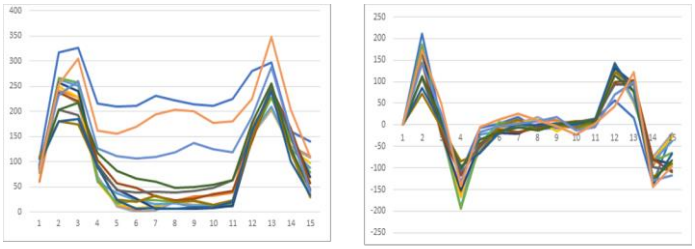
그림 11. 색 발현 판단 영역

색 발현 인식 알고리즘의 설명을 위해서 위의 (그림 11)의 빨간색 사각형 안쪽의 영역에 대하여 색 인식 알고리즘을 적용하였다. Prewitt Mask를 적용하여 변화량을 계산할 경우 3x3 배열로 계산하기 때문에 원본 이미지보다 작은 출력을 가지게 된다. Prewitt Mask를 사용하여 계산한 변화량 값은 (height-2)x(width-2) 크기의 array에 저장한다. height는 원본 이미지의 높이, width는 원본 이미지의 너비이다.

$$Diff(i)=G(i)-G(i-1), i=1,2,\dots,n$$

그림 12. Diff 값 계산 수식

본 논문에서는 색의 영역을 판단하기 위해 색의 경계를 Open Edge와 Close Edge로 나눠서 다룬다. Open, Close Edge를 판단하기 위해서 값이 변화하는 방향을 계산하였다. (그림 12)의 수식을 적용하여 인접한 픽셀과의 변화도 차이, Diff 값을 계산하였다. 앞 column과의 차이를 통해서 변화도의 증감을 표현하였다. 아래의 (그림 13)은 윈도우에 대한 픽셀의 변화량과 Diff값을 그래프로 나타낸 것이다.



(a) (b)

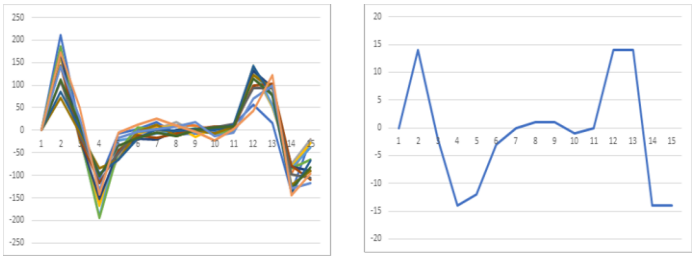
그림 13. (a) Prewitt 적용 값, (b) Diff 값

색의 경계에서는 변화도가 커지므로 Diff값이 커진다. 그러므로 Diff값의 절대값이 15 이상인 경우만 색의 경계로 판단하였다. Diff값에서 색의 경계를 판단하는 식은 아래의 (그림 14)와 같다. 색의 경계를 판단하는 임계값 15는 실험을 통해서 정한 값이다.

$$Edge(i)=\begin{cases}1, Diff(i)\geq 15\\ -1, Diff(i)\leq -15\\ 0, else\end{cases}$$

그림 14. Edge 판단 수식

위의 수식을 통해서 계산된 Edge 배열을 통해 Open, Close Edge의 위치를 판단하기 위해 column별로 Edge의 값을 합하여 Edge Column Sum Array를 생성하였다. Colum Sum을 통해서 판단 영역의 어느 위치에 Open, Close Edge가 존재하는지 판단할 수 있다. 아래의 (그림 15)은 Diff값과 Edge Column Sum Array를 그래프로 나타낸 것이다.



(a) (b)

그림 15. (a) Diff 값, (b) Edge Column Sum Array

(그림 15) (b)의 그래프는 (a) Diff값을 간소화하고 Column별로 합산한 것이므로 유사한 형태를 가진다. 윈도우 상에서 Open Edge는 왼쪽, Close Edge는 오른쪽에 존재하므로 Edge Column Sum Array의 중간을 기

준으로 왼쪽의 최소값(Open Edge), 오른쪽의 최대값(Close Edge)을 계산하였다. 최대값과 최소값의 차이가 이미지의 높이보다 크면 색의 발현으로 판단하였다. 이미지와 실제 값과의 비교를 통해 최대값과 최소값의 차이가 이미지의 높이보다 클 경우 색의 발현으로 인식하는 것이 옳다고 판단하였다.

3.5 노이즈 영역 제거

색 인식 알고리즘을 통해 판단된 결과 중에 노이즈로 인해 색 발현으로 인식되는 경우가 존재한다. 아래의 (그림 16)은 REBA 이미지에 나타나는 노이즈의 예시들이다.

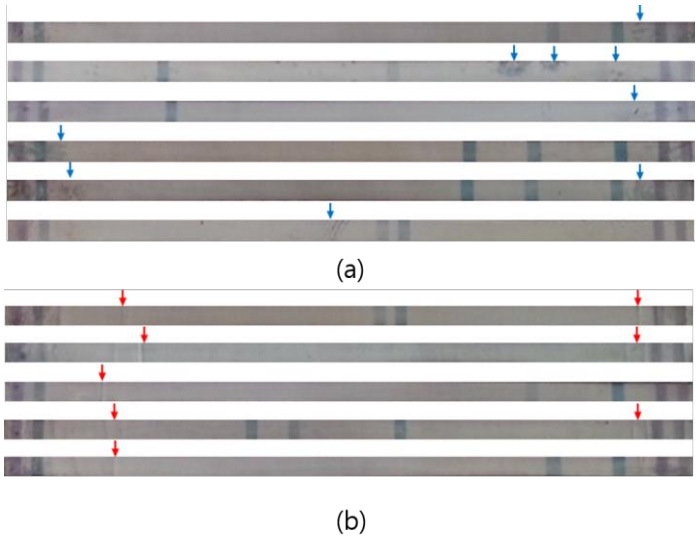


그림 16. (a) 번짐으로 인한 노이즈,
(b) REBA 검사 키트 손상으로 인한 노이즈

REBA 검사 과정에서 미숙련자에 의한 번짐 노이즈, REBA 검사 키트 손상에 의한 노이즈 등등 여러가지 종류의 노이즈가 발생한다. 노이즈로 인해서 색의 발현이 아닌 부분도 색의 발현으로 인식되므로 노이즈 제거 알고리즘을 적용하였다.

REBA 검사 키트의 손상으로 인한 노이즈는 좁은 영역으로 나타나므로 윈도우 상에서 Open Edge와 Close Edge의 거리가 짧게 나타나는 경우에 노이즈로 판단하여 제거하였다. 노이즈로 판별되는 거리는 테스트를 통해서 결정하였으며, 윈도우 너비의 30% 미만일 때 노이즈로 판단한다.

번짐으로 인한 노이즈 같은 경우에는 넓게 나타나고, 색의 분포가 균일하지 않게 나타나는 특징이 있다. 노이즈를 제거하기 위해 Edge Column Sum 값을 사용하였다. Edge Column Sum 값은 윈도우의 각 컬럼에서 Edge가 얼마나 강하게 변화하였는가를 나타내는데, 번짐으로 인한 노이즈의 경우에는 일부 픽셀만 값이 변화하므로 Diff값이 작아진다. Diff값이 작아지면 컬럼 합의 값도 작아지므로 일정 값 미만일 경우에는 노이즈로 판단하였다. 프로그램 테스트를 통해서 이미지 높이의 30% 미만일 경우에는 노이즈로 인한 색의 발현으로 판단하여 제외하였다.

3.6 원인균 판독

REBA 이미지의 32개 영역에 대해 색 인식과 노이즈 제거 알고리즘을 적용하여 색이 발현된 영역을 판단하고, 0과 1의 패턴 조합으로 변환하였다. 패턴은 (그림 17) (a)의 패턴 출력 영역에 색으로 출력한다. 출력 패턴을 3.1절에서 생성한 패턴 테이블과 비교하여 검출된 원인균을 판독한다. 패턴이 일치하게 되면 검출된 원인균의 이름을 모바일 어플리케이션의 화면에 출력한다 (그림 17) (b).

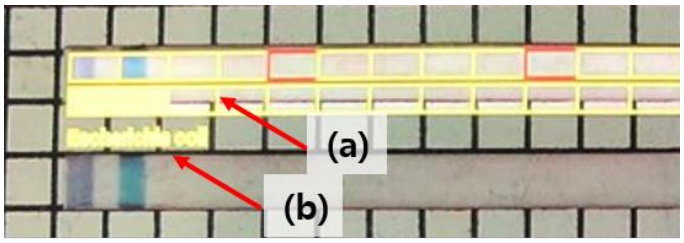


그림 17. 원인균 검출 결과 출력 화면
(a) 패턴 출력 영역, (b) 원인균 출력 영역

4. 실험 및 결과

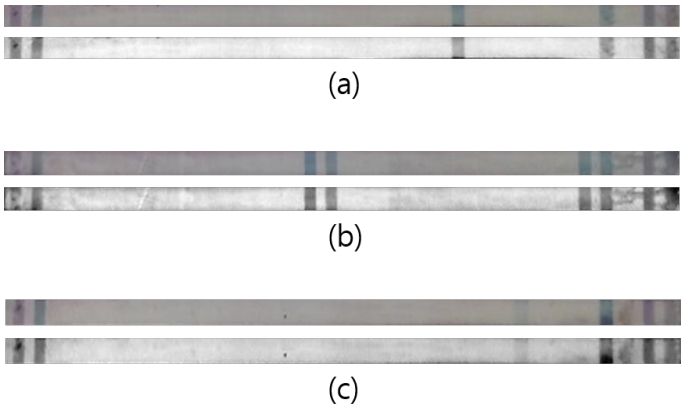


그림 18. (a) 양질의 REBA 이미지, (b) 노이즈 포함 이미지, (c) 흐릿한 색의 발현 포함 이미지

REBA 검사 이미지는 3 종류로 분류할 수 있다. 위의 (그림 18)의 (a)와 같은 노이즈가 존재하지 않고 색의 발현이 뚜렷한 양질의 이미지, (b)와 같은 노이즈가 존재하는 이미지, (c)와 같이 색의 발현이 흐릿하게 나타나는 이미지이다. 어플리케이션의 성능을 검증하기 위해 위와 같은 이미지 들에 대해서 실험을 진행하였다. (a)와 같은 양질의 이미지에 대해 노이즈 제거

알고리즘을 적용하지 않고 판독을 수행한 결과 95% 이상의 판독율을 보였다. (b), (c)와 같은 노이즈가 있거나, 색의 발현이 흐릿한 이미지에 대해 판독을 수행한 결과 판독율이 다소 감소하였다. (b), (c)와 같은 이미지를 제대로 판독하기 위해서 본 논문에서 개발한 노이즈 제거 알고리즘을 적용하여 실험을 수행하였다. 또한, 알고리즘의 성능을 평가하기 위해 기존의 이미지 노이즈 제거 알고리즘과 비교하였다.

이미지상에 존재하는 노이즈를 제거하는 방법인 Mean Filter[14], Median Filter[15], Gaussian Filter[16]를 이미지 전처리과정에 적용하여 본 논문에서 개발한 노이즈 제거 알고리즘과 비교하였다. 아래의 (그림 19)과 (표 2)는 비교를 수행한 결과를 나타낸 것이다. A는 본 논문에서 제안한 노이즈 제거 알고리즘, B는 Mean Filter, C는 Median Filter, D는 Gaussian Filter를 적용한 결과이다.

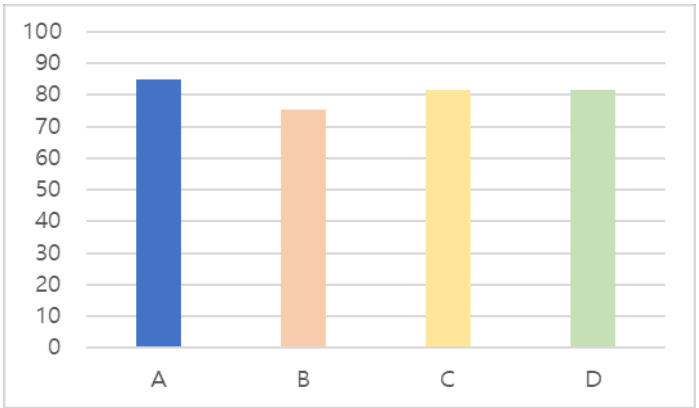


그림 19. 노이즈 제거 알고리즘 판독율 비교(그래프)

표 2. 노이즈 제거 알고리즘 판독율 비교(표)

방법	A	B	C	D
전체	93	93	93	93
판독	79	70	76	76
판독율(%)	84.95	75.27	81.74	81.74

기존의 노이즈 제거 알고리즘 보다 본 논문에서 제안한 노이즈 제거 알고리즘이 높은 판독율을 보이는 것을 알 수 있다. Mean, Median, Gaussian Filter의 경우에는 노이즈 픽셀 주변의 픽셀 값을 참조하여 반영함으로써 노이즈를 제거하는 방법을 사용하기 때문에 이미지의 흐릿한 색의 발현이 배경색과 비슷하게 변경된다. 이로 인해 본래 색의 발현으로 판단될 부분이 배경색으로 판단되어 색 영역 후보군에서 제외된다. 그러므로 REBA 이미지를 변경하지 않고 검출된 색의 영역 후보군에서 노이즈를 판단하여 제거하는 본 논문의 노이즈 제거 방법이 더 효과적이다.

5. 결론 및 향후 연구

REBA 검사 이미지에서 색의 경계를 인식하고 색의 영역을 추출하여 원인균 패턴을 생성하는 알고리즘을 개발하였으며, 생성된 패턴과 원인균 패턴 테이블과 비교하여 REBA 검사를 통해 검출된 원인균을 판독하는 시스템을 모바일상에서 구현하였다. 기존의 패혈증 판독기기들이 PC 및 임베디드 기기 환경에서 판독을 수행하던 것을 벗어나 모바일 환경에서 패혈증 원인균을 판독하는 시스템을 구현하였다.

모바일 어플리케이션을 통해 원인균을 판단하는 방식으로 제안함으로써 패혈증 원인균 판독에 소모되는 시간을 단축하였다. 또한, 육안으로 판단 시 발생할 수 있는 오차에 대하여 어플리케이션을 통한 판독결과를 제공함으로써 연구자가 보다 정확하게 원인균을 판별할 수 있도록 하였다.

REBA 검사와 유사한 방법, 검사키트 상의 색 발현을 통해 병원균을 판독하는 검사 방법에 대해 적용할 수 있도록 할 예정이다. 각 검사 방법 별로 패턴 테이블을 생성하고 색 발현 인식 영역을 지정할 수 있도록 하여 범용적으로 원인균을 인식 및 판독할 수 있는 어플리케이션을 개발할 예정이다.

Mean, Median, Gaussian Filter 등 노이즈 제거 알고리즘들은 이미지에서 색의 경계를 검출하기 전 적용되며, 본 논문의 노이즈 제거 알고리즘은 이미지에서 색의 경계를 검출한 후에 적용되는 등 적용되는 타이밍이 다르다. 본 논문에서 개발한 알고리즘과 기존에 개발된 성능이 우수한 알고리즘들을 결합하여 판독율을 높일 예정이다. 다양한 방식의 노이즈 제거 필터를 적용하면서 REBA 이미지의 변형을 최소화하여 이미지의 노이즈를 제거하는 방식으로 연구를 진행할 예정이다.

Acknowledgement

이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (No. NRF-2014M3C4A7030505)

6. 참고 문헌

[1] Y. Choi et al., PCR-reverse blot hybridization assay for screening and identification of pathogens in sepsis, Journal of Clinical Microbiology, vol. 51, no. 5, pp. 1451-1457, 2013

[2] HY. Wang et al., Real-time PCR TaqMan assay for rapid screening of bloodstream infection, Annals of clinical microbiology and antimicrobials, vol. 13, no. 3, 2014

[3] HY. Wang et al., Multiplex real-time PCR assay for rapid detection of methicillin-resistant

staphylococci directly from positive blood cultures, Journal of Clinical Microbiology, vol. 52, no. 6, pp. 1911-20, 2014

[4] SD. Park et al., Evaluation of PCR-Reverse Blot Hybridization Assay, REBA Sepsis-ID Test, for Simultaneous Identification of Bacterial Pathogens and mecA and van Genes from Blood Culture Bottles, Annals of laboratory medicine, vol. 34, no. 6, pp. 446-455, 2014

[5] HY. Wang et al., Performance of PCR-REBA assay for screening and identifying pathogens directly in whole blood of patients with suspected sepsis, Journal of applied microbiology, vol. 119, no. 5, pp. 1433-1442, 2015

[6] R. Maini and H. Aggarwal, Study and Comparison of Various Image Edge Detection Techniques, International Journal of Image Processing, vol. 3, issue. 1, pp. 1-11, 2009

[7] P. E. Trahanias and A. N. Venetsanopoulos, Color edge detection using vector order statistics, IEEE Transactions on Image Processing, vol. 2, issue. 2, pp. 259-264, 1993

[8] 정차근, 칼라분류와 방향성 에지의 클러스터링에 의한 차선 검출, The Institute of Engineers of Korea-Signal Processing, vol. 48, no. 4, pp. 86-97, 2011

[9] A. N. Evans and X. U. Liu, A Morphological Gradient Approach to Color Edge Detection, IEEE Transactions on Image Processing, vol. 15, no. 6, pp. 1454-1463, 2006

[10] W. Dai and K. Wang, An Image Edge Detction Algorithm Based on Local Entropy, IEEE International Conference on Integration Technology, pp. 418-420, 2007

[11] A. Levin, D. Lischinski and Y. Weiss, A Closed Form Solution to Natural Image Matting, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, issue. 2, pp. 228-242, 2008

[12] X. Chen and H. Chen, A novel color edge detection algorithm in RGB color space, IEEE 10th International Conference on Signal Processing Proceedings, pp. 793-796, 2010

[13] E. Alparslan and F. Ince, Image Enhancement by Local Histogram Stretching, IEEE Transactions on Systems, Man and Cybernetics, vol. smc-11, no. 5, pp. 376-385, 1981

[14] I. Pitas and A. N. Venetsanopoulos, Nonlinear Mean Filter in Image Processing, vol. ASSP-34, no. 3, pp. 573-584, 1986

[15] J. Tukey, Exploratory Data Analysis. Menlo Park, CA: Addison-Wesley, 1977.

[16] R. Jain, R. Kasturi, B. G. Schunck, Machine Vision, McGraw-Hill, 1995

C 언어로 작성된 OSEK/VDX OS 모델을 사용한 임베디드 제어소프트웨어 검증

양송 0 김동우 최윤자
경복대학교 컴퓨터학부
yangsong666@knu.ac.kr, kdw9242@gmail.com, yuchoi76@knu.ac.kr

Verification of embedded control software using OSEK/VDX OS model in C

Song Yang0 Dongwoo Kim Yunja Choi
School of Computer Science and Engineering, Kyungpook National University

요 약

Chung et al[1]. proposed an approach of modeling the OSEK/VDX OS in C language based on OSEK/VDX specification[3], can effectively eliminated the heterogeneity between the underlying operating system and embedded control software as well as proved the feasibility of the approach through extensive experiments and verification. However, the approach in [1] does not involve how the OS model will handle periodic events and how the OSEK/VDX OS model is combined to conduct safety verification of the control software. Therefore, in this article, we will describe how to handle periodic events using alarm from OSEK and how to effectively verify safety of several control software in Erika Enterprise with the OSEK/VDX OS model.

1. Introduction

With the popularity of embedded software, ensure the embedded software runs safely has become an important issue. In order to more accurate verify the safety of embedded software, also the interaction between operating system and control software should be considered while ensuring the safety of underlying operating system and control software logic.

By combining the OS, the interaction safety between the control software and the operating system can be more accurately verified, such as, low priority tasks can preempt high priority tasks. The OS model[1] is based on the OSEK/VDX OS specification [3] and developed by C language. Since the embedded control software is typically written in C language, it removes the heterogeneity between the underlying OS model and the application language. This contribution allows us can focus on how to combine OSEK/VDX model to verify control software safety.

However, the proposed approach in [1] does not fully cover the specification of automotive OS, such as periodic behavior and its 9 out of 26 APIs. Periodic behavior or Alarm is an important object in handling with periodic events, therefore lack of these objects will limit our verification approach. One example is verifying an engine control software with temperature sensor that checks temperature of the engine in every half second. Lack of verifying such system may leads the engine on fire which lead to casualty or enormous property damage to the owner of the car. To solve this issue, we

propose an approach to modeling alarms that mimic the way a real alarm processing. Applying this approach solves the issue that the control software having periodic behavior.

Our approach for modeling alarm is divided into the following steps: first we need to find the cycle time, then we need record the times of context switches by counter, then enter the task schedule when the value of the counter is less than the cycle time, and then execute alarm when the cycle time is equal to the counter.

In addition, in order to support multitasking, it is necessary to model task context switching. We mainly mark the jump position where the task context switch can occur and keep the program counter to each task to understand the progress of the task. This make it possible to jump to the location to resume a task. The infinite loop here is treated as a special case, in order to ensure that other tasks can also perform during the infinite cycle, we place the context switching point in the loop.

Since OSEK/VDX OS configuration files exist separately as input files, OSEK/VDX OS model also needs separate configuration files. We overwrote the object definitions in OSEK/VDX in C language and put them in OSEK/VDX OS model.

Finally, the way how to handle assertions becomes the last issue that needs to be solved. First, we export the attributes from the OSEK specification[3], then we combine the control logic of the actual Erika application to write the assertion statement in C and insert it into the source code.

Figure 1 illustrates the over process of verifying an embedded control software by combining the OSEK/VDX model. First step is to modify the source code. The modification of the source code consists of two parts: 1) inserting the context switch point behind API invoked, and handling the infinite loop by context switching. 2) specifying to assert statements in conjunction with the OSEK specification[3] and execution logic of the control software. The object definitions in the OSEK/VDX configuration file are then exported and rewritten in C. Finally, the alarm is modeled and the annotated source code and configuration files are composed with the OSEK/VDX OS model, meanwhile the properties of the control software are verified using CBMC[12].

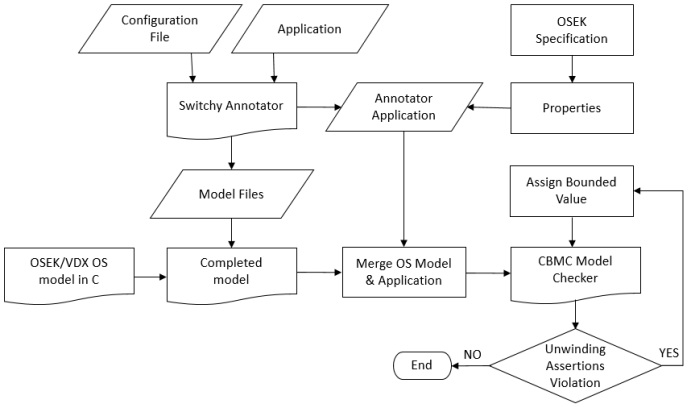


Fig.1. Overall Approach

The remainder of the paper is structured as follow: Section 2 introduces the OSEK/VDX model in C language. Section 3 is a brief description of how to extract configuration information from the configuration file. Section 4 covers context switching in embedded control software source code and how we handle infinite loops. Section 5 discusses ways to model an alarm. Section 6 explains how to specify properties in source code and show the result of verification.

2. Background

2.1 OSEK/VDX OS

OSEK/VDX OS[3] provides a unified environment that offers effective resource management capabilities for automotive electronic control portability. It is widely used as a standard for automotive operating system and IoT devices[10]-[11]. Major characteristics of OSEK OS include: 1) it assumes running on a single processor, and 2) all system variables are required to be statically assigned at the start-up time.

2.2 OSEK/VDX OS model in C language

A number of model-based approaches models OS in formal languages and verifies its safety[8]. However, in most cases, a formal language is used to model operating systems, such as Promela[4], NuSMV[5], and CSP[6], which occurs other problems. When an application is written in C, there is a heterogeneity between the operating system and the application. Thus,

to verify an application code with its OS model, it is necessary to convert the OS model into C or construct a model of an application program in the same language used for modeling the OS.

Therefore, Chung [1] et al. proposed a method of modeling the underlying operating system using C language, to resolving the above problem. This approach mainly uses two steps to model the operating system: first, it defines formal models of OS components in modeling language CSP[6] based on the specifications of international standard OSEK/VDX[3], then defines an OS model in C by using the CSP models as references. By defining the CSP model, OSEK/VDX is divided into five components, including a system model, Task, API Handler, Scheduler as well as the Task context switching. Finally, the attribute list is obtained by combining OSEK/VDX specification[3]. These properties are used to verify the OS model regarding satisfaction of basic requirements of the specification.

2.3 CBMC- Bounded Model Checker for C

The CBMC[12] is a tool for the formal verification of ANSI-C programs using Bounded model checking (BMC)[13]. The CBMC checks the properties defined in the ANSI-C assertion and returns counterexample trace when the assertion is violated. In CBMC, the transition relations of C code and its specification are jointly unwound to obtain a Boolean formula, whose satisfiability is checked using a SAT solver.

3. Configuration file for model

The embedded control software compatible with the OSEK/VDX standard is consists of the two files: configuration file and application code as show in Figure 2. Since the configuration file is written in OIL language[2] as shown in Figure 3, it does not match our modeling language. So we need to export the object properties in the configuration file and rewrite them in C language.

We need two files to initialize the OSEK/VDX model: a C type file which contains the initialization properties of object. The header file contains some global properties for the OS model such as id of each task and the highest priority. They are shown in Figures 4 and 5.

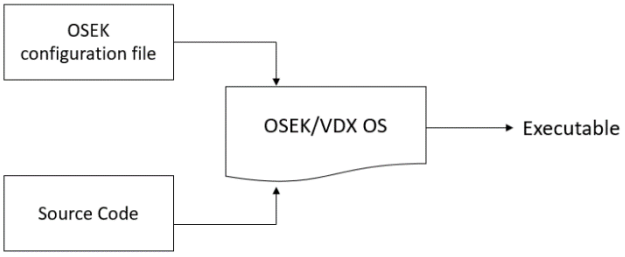


Fig.2. Embedded Control Software File construction

```
TASK Task1{
      PRIOITY = 0x01; /*Low priority*/
      AUTOSTART = FALSE;
      STACK = SHARED;
      ACTIVATION = 1;/*only one pending activation*/
}
```

Fig.3. Task object properties in the configuration file

```
task_static_info[1].max_act_cnt = 10;
task_static_info[1].preemptable = 1;
task_static_info[1].prio = 2;
task_static_info[1].autostart = 0;
task_static_info[1].extended = 0;
task_dyn_info[0].dyn_prio = task_static_info[0].prio;
task_dyn_info[0].act_cnt = 0;
task_dyn_info[1].dyn_prio = task_static_info[1].prio;
task_dyn_info[1].act_cnt = 0;
```

Fig.4. Task object initialization

```
#define MAX_PRIORITY 2
#define Task0 0
#define Task1 1
```

Fig.5. Task object definition

4. Context Switching

4.1 What is Context Switching

OSEK/VDX OS[3] supports multitasking and preemptive scheduling mechanisms. When a particular context switch condition is satisfied, the current task stops executing and scheduler decides whether the task is preempted or not. Example of context switch conditions are as follows:

1) Invoked the APIs that might convert the operating system to an idle state, such as TerminateTask, ChainTask etc.

2) When a task which has a higher priority than the currently running task entered the ready-queue.

3) When a non-preemptive task invokes the Schedule API function.

For example, if preemption condition “a task with higher priority than current running task is in the ready queue.” is satisfied, the higher priority task will preempt the currently running task, which will cause task context switching.

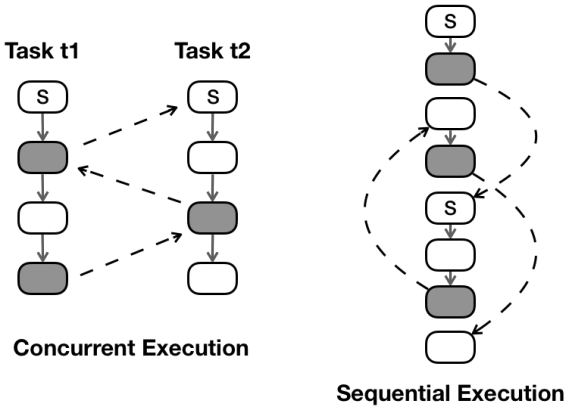


Fig.6. Context Switching Parallel model and Signal Thread model

Figure 6 shows the context switching behavior in the model. In this model, each task has its own thread of execution and the context switching occurs through message passing. For example, when rescheduling point reaches while Task1 is running, it sends a message to the scheduler and waits for a signal to resume the execution. If Task2 has higher priority at that point, the scheduler sends a message to start Task2. In such a case, Task1 waits until Task2 reaches its next scheduling point. On the other hand C model is based on a single thread which is divided into multiple areas. Each area represents a task space and the context switch is modeled as changing the task space using goto statements and other control statements to jump from one task to another.

4.2 Inserting Context Switching point

Figure 7 shows the macro jump_tid needs to implement context switching and the global variable flag used to pass the function return value. Each macro contains global variables, such as current_pc[tid], which contain information on the next context switch points. When a task is preempted, the current_pc[tid] records the preemption point of the task so that it can resume running from the preemption point. When an API function is called, the task gets the return value from the function. The task determines whether to continue or stop based on the return value passed by the flag.

```
int flag;

#define jump_1(){\
    switch(current_pc[1]){\
     case 1:\
          goto lable_1;break;\
     case 2:\
          goto lable_2;break;\
     case 3:\
          goto lable_2;break;\
     }\
}
```

Fig.7. Macro for context switching

Context switching can occur after some API calls. The kind of APIs that require rescheduling is described in OSEK/VDX OS specification[3]. Figure 8 shows an instance of an application source code before and after inserting context switching point, which needs to be used in conjunction with the code in Figure 7. When the ActivateTask API is invoked, the scheduler in the OS model to determine whether the current task continues to execute. If Task2 has higher priority, then Task1 is preempted and current_pc[1] is updated immediately. When the task resumes running, it will rerun from the location of label_2.

```
void TASK(1) {
     ActivateTask(2);
}
```

```
void TASK(1) {
     jump_1();
label_1:
     flag = ActivateTask(2);
     current_pc[1]++;
     if (flag) return;

label_2:
}
```

Fig.8. Explicit model of context switch

4.3 Handling Infinite loop

Embedded control software often contains infinite loop. These loops are usually only used to wait for the execution of an alarm or other function. When using CBMC[12] of verification an embedded control software, it is impossible to detect statements outside the loop due to the existence of an infinite loop.

To solve the issue caused by infinite loop, we use an approach of inserting context switch point in infinite loop. Figure 9 shows the context switching behavior in an infinite loop. When the loop reaches the context

switch point, the loop aborts execution and waits for the resumption of execution, during which the high-priority task can preempt the execution of the current task. Thus, solving the issue that other tasks cannot execute is due to the infinite loop.

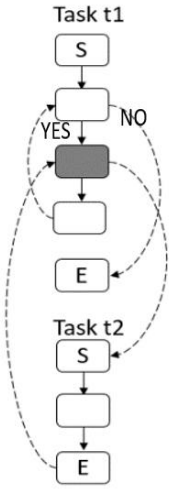


Fig.9. Context switching model in an infinite loop

```
void TASK(1) {
    while(1){;}
}
```

```
void TASK(1) {
    jump_1();
Label_1:
    while(1)
    {
        current_pc[1]++;
        return;
        label_2:
        ;
    }

}
```

Fig.10.Explicit model of infinite loop context switching

Figure 10 shows an instance before and after insert context switching point, which needs to be used in conjunction with the code in Figure 7. This approach is similar to the approach in 4.2. When the task enters the loop, it updates the value in current_pc[tid] immediately, then the loop aborts execution and waits for the resumption of execution. During this period, other high-priority tasks can preempt the execution of the current task, continue execution from the position of label_2 when the task resumes.

In addition, a scheduling function called App is the core of implementing context switching as shown in Figure 11. The order in which tasks is executed is passed by the scheduler.

5.Handling alarm

Alarm is an object in OSEK/VDX for processing periodic events[3]. Figure 12 shows alarm in the configuration file as an OSEK/VDX object. There are

```
void app()
{
    while(current_tid >= 0)
    {
        if(current_tid == 0)
            TASK(0);
        else if(current_tid == 1)
            TASK(1);
        else if(current_tid == 2)
            Task(2)
    }
}
```

Fig.11. Scheduler app function

several options in the Figure: COUNTER, ACTION AUTOSTART. The ACTION option is responsible for the behavior that the alarm will perform, and the AUTOSTART option is used to set whether the alarm is automatically started. Beyond that, there are two sub-options in AUTOSTART: ALARMTIME refers to the time when the alarm is executed for the first time, CYCTIME is the periodic execution time of alarm. Since the alarm is automatically start in this case, it will be started along with the system.

```
ALARM alarm{
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK{ TASK = task1;}
    AUTOSTART = TRUE{
        ALARMTIME = 20; CYCTIME = 10; APPMODE = std;
    }
}
```

Fig.12. Alarm object properties in the configuration file

The basic idea of our approach to modeling alarms is to insert a function named alarm to the source code. The execution order between the task and the alarm is determined by a scheduler function named App. We insert the alarm function in the source code. The API invoked in the alarm function is specified by the configuration file, as shown in Figure 12. In the case of Figure 12, the alarm is used to periodically activate Task1, thus, we have to put the API function ActivateTask(task1) in the alarm function. After that, we

will handle the scheduler function app. We insert the part of the alarm schedule in the schedule function app and also specify a value called CYCTIME in the scheduler function App, which is provided by the parameters in SetRelAlarm(). If the alarm is automatically started, the named CYCTIME value can be found in the OSEK/VDX configuration file. Thereafter, a counter is inserted to the scheduling function APP(), which the initial value is 0. The value of the counter increments by one in every times loop completed. When the value of the counter is less than CYCTIME is going to enter the Task schedule, and an alarm is executed when the counter is equal to CYCTIME. In addition, a function called alarm_is_on is placed before the alarm is executed to checks the alarm status and will return 0 or 1. If alarm is activated it will return 1, otherwise, return 0. This ensures that the alarm will not be executed until the alarm is activated.

In addition, let us briefly introduce the conditional statement of the while loop in the APP function. There is currently a task that needs to be executed or the alarm is activated in the current, when satisfy one of them will enter the loop. Such a conditional statement guarantees that a loop will not jump out as long as a task needs to be executed. The second conditional statement guarantees that the alarm can also run normally when there are no tasks. The above steps are shown in Figure 13.

6. Experiments

6.1 Property Specification

In order to record the times of executions of the alarm and task, we inserted three types of counters in the source code, which are task_tid_fired, task_tid_ended and alarm_counter. The task_tid_fired counter is automatically increments by 1 when the task is successfully activated. Similarly, the task_tid_ended counter will be automatically increment by one when the task terminates. The alarm_counter is placed in the alarm function to record the times of the alarm execution.

Table 1 shows the general properties of control software. We apply the following assertion statements in conjunction with each control software's own situational properties.

6.2 Erika specific functionality

Erika Enterprise is a free of charge, open-source RTOS implementation of the ISO 17356 API (derived

```
alarm_is_on(){
{
    if(alarm_state[alarm] != CANCELED)
        return 1;
    else
        return 0;
}

void alarm(){
    ActivateTask(2);
}

int CYCTIME = 10;
int counter = 0;
void app()
{
    while(current_tid >= 0 || alarm_is_on() == 1)
    {
        counter++;
        if(counter < CYCTIME)
        {
            if(current_tid == 0)
                TASK(0);
            else if(current_tid == 1)
                TASK(1);
        }
        else if(counter == CYCTIME)
        {
            if(alarm_is_on() == 1)
            {
                alarm();
                counter = 0;
            }
        }
    }
}
```

Fig. 13. Completed alarm model

from the OSEK/VDX API) OSEK/VDX certified[8]. In this study, the six Erika x86 monostack applications are taken as instance and safety verification is carried out In combination with OSEK/VDX OS model. The main function in the embedded control software is executed as a task in the OSEK/VDX OS. Therefore, we made a small modification of the model os that no longer use task 0 as the label for all tasks terminated. We changed this tag to task -1, so that we can add a new task

named Task0 to the application, that is, Task 0, the start-up mode is AUTOSTART and the priority is 0 which is responsible for running the text segment of the main function. In addition, the main function is recreated to contain only one named StartOS, which is used to start the OSEK/VDX OS.

Table.1. Assertion of embedded control software in OSEK/VDX model

No	Semantic	Property
A1	Higher priority task preempts lower priority task	task_static_info[tid].prio <task2_static_info[tid].prio && task_tid_fired >0 && task_tid_ended ==0 ? task_state[tid]== Running:1
A2	The task that obtains the resource will not be preempted by other tasks	Resource_Table[Rid].alloc == tid && task_tid_fired > 0 ? !(task_state[tid] == Running):1
		Resource_Table[tid].alloc == 0 && task_tid_fired > 0 ? (task_state[tid] == Running):1
A3	Verify that the task activates and terminates normally	task_tid_fired > 0;
		task_tid_ended > 0;
A4	Activated alarm fires its periodic behavior	task_tid_fired > 0;
		alarm_counter > 0 ;

6.3 Experimental environment and tools

All experiment were performed with Linux Fedora-release -19 x86_64 architecture, 8 Intel(R) Xeon(R) CPU E31280 @ 3.50GHz, and 32GB of total memory, CBMC 5.10 version.

Before the verification starts, we need to specify a loop unwinding value of CBMC[12]. If the loop unwinding value is too small, the unwinding assertions violation occurs due to insufficient unwinding of the loop. When unwinding assertions violation occurs, we have to re-specify the loop unwinding value until the unwinding assertions violation disappears or the execution time of the verification is unacceptable.

Since our approach requires modifying the source code and rewriting the configuration file, which increases the cost of verification to a certain degree. Therefore, in order to reduce the cost of modifying embedded control software, we developed an automation tool called Switchy[7], it automatically generates the required files for the OSEK/VDX model and annotates the source code from configuration files.

6.4 Verification Result

Due to the alarm is executed by indefinitely, it will caused great distress for our verification, because it would trigger CBMC[12] unwinding assertions violation, if the verification results included unwinding assertion violation then verification results will be not credible, so we adopted to reduce the times of alarm to solve verification results will unwinding assertion violation issue.

Table.2. Verification performance of properties

APPs	property	VCC	Time (Sec)	MEM (MB)
EE_test03	A1,A3	3,092	5.5	174
EE_test04	A1,A3,A4	6,300,314	7m17	25,150
EE_test05	A1,A3,A4	6,481,342	6m45	27,930
EE_test06	A1,A3,A4	2,008	6.7	534
EE_test07	A1,A3,A4	8,534,131	7m	28,750
EE_test08	A1,A2,A3,A4	8,255,115	5m14	24,170

We performed extensive validation with each control software using the expressions described in Table 1. Due to performance limitations of the devices we use for verification, we finally limited the alarm to 24,900 times, and unwinding value was specified as 250,000. Table 2 shows the properties performance.

As shown in table 2, all control software verification was completed without unwinding assertions violation within 10 minutes. Since EE_test03 and EE_test06 do not contain alarm, their verification costs are less.

Due to performance limitations of the devices we use for verification, we were unable to verify more times of alarm execution than 24900. We can only draw conclusions based on the current situation: when the times of alarm executions is less than or equal to 24,900 and the number of loops unwind is less than or equal to 250,000 times, our approach can be used to effectively safety verify the embedded control software.

7. Conclusions

We propose an approach to verify embedded control software by combining OSEK/VDX OS model. This

approach consists of inserting context switch points and assertion statements in the source code, modeling the alarm, and generating configuration files of OSEK/VDX OS model. Finally, CBMC[12] model checker is used to verify the safety of the control software.

Using our approach combined with OSEK/VDX OS model to verify the six of embedded control software. Our approach will make a series of changes to the control software and reduce the times of alarm executions to effectively safety verify the control software without triggering unwinding assertions violations.

Since our OSEK/VDX OS model does not have an ISR model, the control software including ISR cannot be accurately verified. In the future work, we will gradually improve the complete functions in all OSEK/VDX OS including ISR.

Acknowledge

This research has been supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-2016R1D1A3B01011685), and by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science, ICT (2017M3C4A7068175).

References

[1] Yoohee Chung, Dongwoo Kim and Yunja Choi. “Modeling OSEK/VDX OS Requirements in C.” 2017 24th Asia-Pacific Software Engineering Conference (APSEC) (2017): 398-407.

[2] Zahir, Andree. "OIL-OSEK implementation language." (1998): 8-8.

[3] OSEK Group. "OSEK/VDX Operating System Specification." site web: http://www. osek-vdx. org (2005).

[4] Holzmann, G. J. "Promela language reference." Bell Labs(1997).

[5] Cimatti, Alessandro, et al. "Nusmv 2: An opensource tool for symbolic model checking." International Conference on Computer Aided Verification. Springer, Berlin, Heidelberg, 2002.

[6] Hoare, Charles Antony Richard. "Communicating sequential processes." Communications of the ACM 21.8 (1978): 666-677.

[7] 정유희, 최윤자: “OSEK/VDX 기반 내장형 소프트웨어 검증을 위한 환경 설정 추출 및 소스코드 자동 변환 도구”, 한국소프트웨어공학 학술대회 논문집, 20 권, 1 호, pp . 96-99 2018.

[8] Kang, Eun-Young, Gilles Perrouin, and Pierre-Yves Schobbens. "Model-based verification of energy-aware real-time automotive systems." Engineering of Complex Computer Systems (ICECCS), 2013 18th International Conference on. IEEE, 2013.

[9] “Erika enterprise.” http://erika.tuxfamily.org 2002

[10] Zahir, Andree, and Patrick Palmieri. "OSEK/VDX-operating systems for automotive applications." (1998): 4-4.

[11] Yunja Choi, "Automated Validation of IoT Device Control Programs Through Domain-Specific Model Generation." International Conference on Software Engineering and Formal Methods. Springer, Cham, 2018.

[12] Kroening, Daniel, and Michael Tautschnig. "CBMC-C bounded model checker." International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, Berlin, Heidelberg, 2014.

[13] Biere, Armin, et al. "Bounded model checking." Advances in computers 58.11 (2003): 117-148.

모델 기반 멀티타스크 프로그램의 API 호출 안전성 검증을 위한 오경보 식별 및 성능 향상 기법[1]

김동우 최윤자
경북대학교 컴퓨터학부
Kdw9242@gmail.com yuchoi76@knu.ac.kr

False-alarm Detection and Performance Improvement in Model-based API-call Safety Checking for Multitask Programs

Dongwoo Kim, Yunja Choi
School of Computer Science and Engineering, Kyungpook National University

요 약

멀티타스크 프로그램의 모든 문맥 전환을 고려한 엄밀한 검증은 운영체제의 스케줄링 정책에 반하는 시나리오가 보고될 수 있다는 단점이 있다. 반면 운영체제 모델과 추상화된 어플리케이션 모델을 구성하여 모델 기반 검증을 수행할 경우 스케줄링 정책을 준수하는 시나리오 만을 분석한다는 장점이 있으나 어플리케이션 프로그램의 추상화로 인한 오경보가 발생할 수 있다는 단점이 있다. 본 연구에서는 코드 검증 기법을 적용하여 모델 기반 검증 결과 보고된 오류 시나리오의 오경보를 식별하는 방법을 소개하며 식별 시간을 최소화하기 위한 성능 향상 방법들을 소개한다.

1. 서 론

멀티타스크 프로그램은 다수의 타스크가 운영체제의 스케줄링 정책에 맞게 번갈아 수행되는 프로그램이며 운영체제는 타스크의 우선순위와 리소스 소유 여부 등을 분석하여 다음 실행할 타스크를 선정한다. 멀티타스크 프로그램을 검증하기 위해서는 운영체제의 행위를 고려한 검증을 수행해야하며 그렇지 못할 경우 우선순위가 낮은 타스크가 먼저 실행되거나 비선점형 타스크가 선점되는 등 오경보 시나리오가 보고된다.

선행연구[1] 에서는 멀티타스크 프로그램 검증을 위해 운영체제의 행위를 상태 기계 형식의 정형 모델로 정의하고 어플리케이션 프로그램의 분기문과 데이터 타입을 추상화하여 상태 기계 모델을 생성하여 API 호출 안전성 검증을 수행하였다. 검증 결과 안전성 위반 시나리오가 있는 경우 항상 스케줄링 정책에 부합한 시나리오가 보고되었지만 분기문 추상화로 인해 실행 불가능한 시나리오가 보고될 수 있었다.

본 연구에서는 코드 검증 기법을 이용하여 모델 기반 검증 결과 보고된 시나리오의 오경보를 식별하는 방법을 소개하며 식별 과정에서 발견된 오경보 여부를 사용자에게 알려주고자 한다. 또한, 세 가지 성능 향상 방안을 제시하였으며 식별 시간을 평균 89.205초에서 0.199초로 단축시킬 수 있음을 확인하였다.

[1] 이 논문은 2018년도 정부(과학기술정보통신부, 교육부)의 재원으로 한국연구재단-차세대정보 컴퓨팅기술개발사업의 지원을 받아 수행된 연구임. (2017M3C4A7068175, NRF-2016R1D1A3B01011685)

2. 오류 시나리오의 오경보 식별

모델 검증 결과 보고된 시나리오가 프로그램 실행 과정에서 발생 가능하지 않는 경우 오경보이다. 오경보 식별은 시나리오에 나타난 행위가 프로그램에서 나타날 수 있는지 확인하는 과정이다. 모델 검증 결과 보고된 시나리오는 상태 시퀀스로 구성되어 있으며 각 상태는 실행 타스크과 실행 타스크의 API 호출로 구성되어있다. 오경보 식별을 위해서는 시나리오와 동일한 순서로 타스크를 실행할 수 있으며 동일한 순서로 API를 호출하는 실행 경로의 존재 유무를 확인하여야 한다.

오경보 식별을 위해 프로그램에 존재하는 모든 실행 경로가 탐색되어야 한다. 탐색 대상 실행 경로는 프로그램의 분기 조건을 고려했을 때 실행 가능하여야 하며 운영체제를 고려했을 때 스케줄 가능해야한다. 본 연구에서는 프로그램의 분기 조건을 모두 고려하여 실행 가능한 경로만 탐색하기 위하여 CBMC 코드 검증 도구[2]를 사용하였다.

CBMC는 assert 속성 검증 도구이며 프로그램에 존재하는 모든 실행 경로를 탐색하여 assert 함수의 매개 변수로 사용된 조건을 만족하지 않는 실행 경로를 찾는다. 또한 C 프로그램 코드 자체를 입력 받고 별도의 추상화를 요구하지 않으며 프로그램에 존재하는 모든 분기문을 고려하여 실행 가능한 경로만 탐색한다.

CBMC를 이용하면 실행 가능한 경로를 모두 탐색할 수 있으나 운영체제 정보가 없으므로 스케줄 가능하지 않은 경로도 함께 탐색된다. 탐색되는 실행 경로의 스케줄 가능성을 확인하기 위해 운영체제 정보가

포함된 운영체제 프로그램이 함께 고려되어야 한다. 하지만 CBMC는 프로그램 크기에 따라 기하급수적으로 많은 분석 시간을 요구하므로 큰 프로그램을 추가할 경우 높은 성능 저하가 뒤따르게 된다. 본 연구에서는 운영체제를 사용하지 않고 시나리오에 나타난 정보를 이용하여 멀티타스크 프로그램을 하나의 타스크로 실행할 수 있는 시퀀셜 프로그램으로 변경함으로써 검증 비용을 낮추었다.

시퀀셜 프로그램을 구성하기 위해 본 연구에서는 시나리오 드라이버와 문맥 전환 매커니즘을 정의하였다. 시나리오 드라이버는 시나리오에 나타난 실행 타스크를 순서대로 활성화하는 프로그램 함수이다. 또한 시나리오 드라이버의 맨 마지막에는 assert(false)를 삽입함으로써 CBMC가 시나리오에 나타난 타스크를 순서대로 실행할 수 있는 실행 경로를 보고하도록 하였다. 호출된 타스크는 시나리오에 따라 다른 타스크에게 제어를 넘겨주어야 할 때가 있다. 이를 위해 각 타스크가 API를 호출할 때마다 제어를 드라이버에게 반환하도록 하였으며 드라이버는 시나리오를 확인하고 다음 실행할 타스크에게 제어를 넘겨주도록 하였다. 제어를 반환한 타스크는 재활성화 되었을 때 타스크의 시작이 아닌 이전 실행 지점부터 수행되어야 하므로 각 타스크마다 프로그램 카운터 변수를 두어 제어를 반환할 때 현재 프로그램 위치를 저장하도록 하고 제어를 다시 얻었을 때 복원하도록 하였다.

호출된 타스크는 시나리오에 나타난 API 호출 외 다른 API 호출도 수행할 수 있으나 다른 API 호출을 수행하는 실행 경로는 관심대상이 아니므로 탐색 대상에서 제외할 필요가 있다. 이를 위해 CBMC의 assumption을 사용하였으며 assumption은 매개변수로 제공된 조건이 만족되지 않는 실행 경로를 탐색 대상에서 제외한다. 매 API 호출마다 호출된 API가 시나리오에 나타난 API와 동일함을 확인함으로써 시나리오에 나타난 API 호출 순서와 동일한 순서로 API를 호출하는 실행 경로만 탐색하였다.

CBMC는 시나리오와 동일한 순서로 API를 호출하는 실행 경로를 찾으면 보고할 것이며 그렇지 않을 경우 존재하지 않음을 알려줄 것이다. 존재하지 않을 경우 오경로로 식별할 수 있다.

3. 실험

3.1 실험 구성

오경보 식별 기법의 효과를 확인하기 위해 python 언어의 regex 라이브러리를 이용하여 이를 구현하였으며 선행 연구[1]에서 API 호출 제약사항 “타스크가 대기 상태가 될 경우 언젠가는 대기 상태를 빠져나와야 한다”를 검증한 결과 보고된 오류 시나리오의 오경보를 식별하였다. 모든 실험은 3.4-GHz Intel Xeon E5-1680 CPU와 128 GB RAM을 가진 Fedora Linux에서 실행되었다.

3.2 오경보 식별의 성능 향상 방법

본 연구에서는 세 가지 오경보 식별 성능 향상 방안을 적용하였으며 첫 번째로 타스크의 재실행여부에 따라 프로그램 카운터를 초기화하였으며 두 번째로 드라이버에서 각 타스크를 호출할 때 사용하는 루프를 제거하고 시나리오 길이만큼 타스크를 활성화하였으며 세 번째로 드라이버에서 사용되는 배열의 크기가 충분하지 않을 경우 검증 시간이 낭비되므로 CBMC의 bounds-check로 배열의 크기가 충분함을 확인하였다.

표 1. 테스트 스윗 검증 및 오경보 식별 결과

App	T	LoC	Verif. (s)	Scen.	Reach. w/o Opt.		Reach. w. Opt.	
					Time(s)	F	Time(s)	F
1	6	172	40	4	0.11	O	0.131	O
2	5	329	195	28	0.16	X	0.145	X
3	6	409	363	34	0.82	X	0.350	X
4	5	419	105	18	33.50	X	0.158	X
5	5	419	67	20	0.21	O	0.224	O
6	6	305	593	25	0.17	X	0.168	X
7	6	505	184	7	3.34	X	0.173	X
8	6	317	101	25	0.21	X	0.216	X
9	5	362	436	8	4.32	X	0.153	X
10	7	346	762	32	0.23	X	0.277	X
11	6	571	143	6	3.29	O	0.118	O
12	6	276	157	22	0.16	O	0.211	O
13	6	310	97	26	626.84	X	0.229	X
14	6	308	96	26	575.51	X	0.236	X

3.3 실험 결과

표 1은 검증 결과 및 오경보 식별 결과를 나타내며 각 열은 프로그램 ID, 타스크 수, 코드 수, 모델 기반 검증 시간, 보고된 시나리오의 길이, 성능 향상 적용 유무에 따른 오경보 식별 소요 시간과 시나리오의 실행 가능 여부를 나타낸다. 실험에 사용된 14개의 시나리오 중 4개는 동일한 실행 경로를 프로그램에서 탐색하였고 나머지는 탐색하지 못하였으며 수작업으로 분석한 결과와 일치하였다. 성능 향상 기법 적용 결과 식별 시간을 평균 89.205초에서 0.199초로 단축하였다.

4. 결론 및 향후 연구

본 연구는 모델 기반 검증 결과 보고된 시나리오의 오경보 식별 방안과 성능 향상 방안들을 소개하였으며 실험을 통해 빠른 시간 내에 식별 가능함을 보였다. 향후에는 오경보 시나리오를 모델 정제에 이용할 것이다.

참고 문헌

[1]. Kim, Dongwoo, and Choi, Yunja . "A two-step approach for pattern-based API-call constraint checking." Science of Computer Programming 163, 19-41, 2018

[2]. Clarke, Edmund, Daniel Kroening, and Karen Yorav. "Behavioral consistency of C and Verilog programs using bounded model checking." Proceedings of the 40th annual Design Automation Conference. ACM, 2003.

코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 향상

최현우, 유 신
한국과학기술원 소프트웨어대학원, 한국과학기술원 전산학부
hw.choi@kaist.ac.kr, shin.yoo@kaist.ac.kr

Improving Fault Localization Based on Neural Network Using Change Metrics

Hyeonwoo Choi, Shin Yoo
KAIST Software Graduate Program, KAIST School of Computing

요 약

결함 위치 추정은 소프트웨어 테스팅 과정 중 실패한 테스트 케이스가 있을 경우, 이의 원인이 되는 프로그램 코드의 결함 위치를 찾는 과정으로 사람이 직접 수행하기에는 많은 시간과 비용이 소모를 필요로 한다. 결과적으로 사람의 개입을 최소한으로 하면서 프로그램 결함의 위치를 찾는 기술에 대한 연구가 진행되었으며, 그 중 신경망 기반 결함 위치 추정 기법은 프로그램의 커버리지와 테스트 결과를 사용하여 프로그램 결함의 위치를 추정한다. 그래서 신경망 기반 결함 위치 추정에서 결함과 관련 있는 양질의 데이터를 학습 데이터로 제공하는 것이 중요하다. 정확성 향상을 위해서 본 연구는 코드 변경 지표를 추가한 신경망 기반 결함 위치 추정 기법을 제안한다. 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법은 코드의 결함과 직접적인 영향을 미치는 코드 변경 지표를 추가함으로써 신경망 기반 결함 위치 추정의 정확성을 향상시킨다. 본 연구에서 제안한 기법의 정확성을 측정하기 위해서 Deject4J내의 4개의 프로그램에 대해서 기존의 신경망 기반 결함 위치 추정 기법과 제안한 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법에 대해 실험을 수행하였다. 본 연구에서 제안한 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법은 기존의 신경망 기반 결함 위치 추정보다 0.1% ~ 7.15% 결함을 더 찾아 결함 위치 추정의 정확성을 높였다.

1. 서 론

오늘날 소프트웨어의 규모와 복잡성 증가로 인해 소프트웨어 결함을 최소화 하기 위해 소프트웨어 테스트의 중요도가 증가하고 있다. 소프트웨어의 규모와 복잡성 증가로 인하여, 사람이 직접 결함 위치 추정을 수행하는 것은 소프트웨어 테스팅 과정 중 많은 시간과 비용이 소모된다. 따라서, 사람의 직접적인 개입을 최소화 하면서 프로그램 결함의 위치를 알려주는 효과적인 결함 위치 추정 기술에 대한 연구[1]되어 왔다. 다양한 결함 위치 추정 기법 중 인공지능 및 신경망으로 인해 많이 연구된 기법은 신경망 기반 결함 위치 추정 기법이다.

신경망 기반 결함 위치 추정은 테스트 케이스의 실행으로 얻은 커버리지-테스트 결과 쌍을 이용하여 프로그램의 각 구문 결함 의심도를 계산하여 소프트웨어 개발자가 결함을 찾도록 도와준다. 신경망 기반 결함 위치 추정의 아이디어는 각 테스트 케이스의 실행으로 얻은 커버리지-테스트 결과를 신경망에 이용하여 것으로 실패하는 테스트 케이스들이 자주 커버하는 구문이 결함이 가능성이 높으며, 성공하는 테스트 케이스들이 자주 커버하지 않는 구문이 결함 가능성이 높다. 기존의 신경망 기반 결함 위치 추정 기법은 테스트 실행의 커버리지 및 결과를 가지고 결함 위치 추정을 하게 된다.

다른 연구들에서는 기존의 신경망 기반 결함 위치 추정 기법의 성능을 높이기 위해 신경망의 노드의 수를 변경

또는 신경 망의 계층의 수를 늘려 성능을 향상시키려는 노력을 하였다. 하지만, 다른 연구들은 코드의 직접적인 영향을 미치는 요소들을 변경한 것이 아닌, 간접적인 신경망 노드의 수나 계층 수를 변경하여 성능을 높이려 하였다. 간접적인 요소의 변경은 성능의 한계가 있다. 또한, 기존의 연구와 다른 연구들은 실험 대상 프로그램을 실제 생활에서 사용하며 규모가 큰 프로그램이 아닌, 실험을 위해 임의로 만든 작은 토이 프로그램을 사용하였다.

본 연구에서는 간접적인 요소의 변경의 한계를 극복하고 성능 향상을 위해 코드에 직접적인 영향을 미치는 정보를 추가하고 임의의 작은 토이 프로그램이 아닌, 실제 생활에서 사용하는 규모가 큰 프로그램에 실험 대상을 적용한 결함 위치 추정 기법을 제안한다. 본 연구의 기법은 신경망 기반 결함 위치 추정 기법에 코드와 직접적으로 영향을 미치는 코드 변경 지표를 사용하여 정확성을 향상 시킨다. 코드 변경 지표는 코드의 버전 관리 내역을 이용하여, 코드가 언제 작성 및 수정되었는지, 얼마나 빈번하게 수정 되었는지에 대한 코드 변경 지표로, 작성 및 수정이 오래된 코드와 덜 빈번하게 수정된 코드는 많은 테스트를 거쳐 코드 결함이 낮은 반면 최근 작성 및 수정된 코드와 빈번하게 변경된 코드는 오래되고 코드와 덜 빈번하게 수정된 코드에 비해 테스트를 적게 거쳐 결함이 클 가능성이 높다. 본 연구는 신경망 기반 결함 위치 추정 기법에 코드가 언제 작성 및 수정 되거나, 얼마나 빈번하게 수정 되었는지에 대한 코드 변경 지표를 사용하여 결함의 정확성을 향상 시킨다. 또한, 본 연구는 기존 연구들에 적용한 작은 토이 프로그램이 아닌, 실제 생활에서 사용하고 규모가 큰 프로그램에 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정의 기법을 적용하였다. 본 연구에서 제안한 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법의 효과성 및 확장성을 예측하기 위해서 다음과 같은 연구 질문을 지정하였다.

RQ1. 기존의 신경망 기반 결함 위치 추정과 비교했을 때, 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법은 얼마나 효과적인가?

RQ2. 기존의 실험을 위해 임의로 만든 작은 토이 프로그램에서, 실제 생활에서 사용하고 규모가 큰 프로그램에 적용이 가능한가?

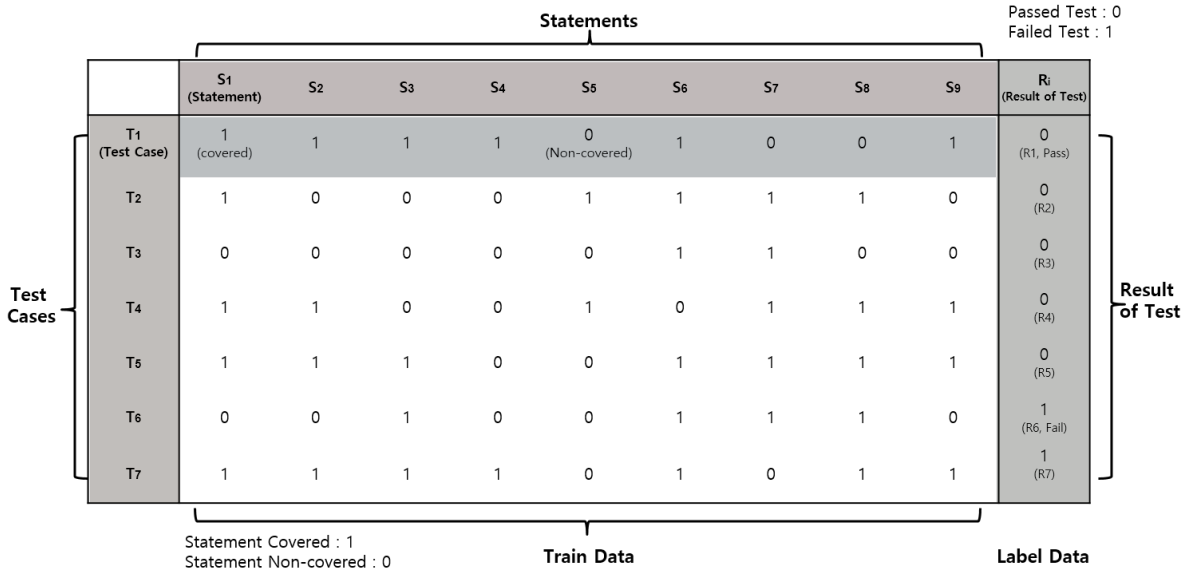
본 연구는 위의 연구 질문들에 대한 답을 위해 신경망 기반 결함 위치 추정과 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법을 실제 생활에서 사용하는 규모가 큰 Defects4J의 Chart, Lang, Math, Time 4개의 프로그램에 적용하고 해당 기법의 정확성을 측정하였다. 4개의 프로그램 모두 실제 생활에서 사용하는 규모가 큰 프로그램들이다. 본 연구의 실험 결과 코드 변경 지표를 이용한 신경망 결함 위치 추정 기법이 기존의 신경망 결함 위치 추정 기법 보다 더 효과적으로 결함 구문의 위치를 추정한다는 것을 보여준다. 기존의 기법과 비교했을 때, 평균적으로 2.96%더 정확하게 결함 위치를 추정한다. 본 연구의 연구 성과는 다음과 같다. 코드 변경 지표를 사용하여 신경망 기반 결함 위치 추정 기법에 직접적인 코드의 결함 정보를 주어 기존의 신경망 기반 결함 위치 추정 기법보다 더 효과적인 기법을 제시하였다. 또한 기존 연구들의 규모가 작으며 실험을 위한 임의의 토이 프로그램인 실험 대상에서, 규모가 크며 실제 생활에서 사용하는 프로그램으로 확장 및 적용 가능성을 증명하였다. 해당 논문의 나머지 구성은 다음과 같다. 2장에서는 신경망 결함 위치 추정 기법과 관련된 연구들에 대해서 설명하고 본 연구 기법과 다른 위치 추정 기법들을 간단히 비교한다. 3장에서는 본 연구에서 제안한, 코드 변경 지표를 이용한 새로운 신경망 기반 결함 위치 추정 기법에 대해서 설명한다. 4장에서는 실제 생활에서 사용하는 Defects4J의 4개의 프로그램을 대상으로 수행한 실험의 환경 및 실험 설계에 대해서 설명한다. 5장에서는 실험 결과를 제시하고 분석하며, 6장에서는 본 논문의 결론과 향후 연구 방향에 대해서 논의 한다.

2. 관련 연구

신경망 기반 결함 위치 추정을 제안한 이후 많은 기법들이 결함의 위치 추정 정확성을 향상시키기 위한 많은 연구들이 진행되었다. 신경망 기반 결함 위치 추정 기법

은 테스트들이 커버한 커버리지와 테스트 결과를 이용

그림 1 신경망 학습을 위한 학습 데이터 구조



법들이 제안되었다.

하여 각 구문의 의심도를 계산하고 결함의 위치를 추정한다. Zheng과 동료들[3]은 신경망의 노드의 개수와 계층을 늘려 신경망 기반 결함 위치 추정의 정확성을 향상시키고자 하였다. 그들은 기존의 입력, 히든, 출력 레이어로 구성된 부분에서 히든 레이어를 여러 개로 늘렸다. 그리고 신경망의 노드의 수를 늘려 신경망을 더 깊게 학습하여 결함 위치 추정의 정확성을 높였다. 이 기법은 실질적인 결함에 큰 영향을 미치지 않았지만, 간접적으로 신경망의 레이어와 개수를 여러 개로 늘려 결함 추정의 정확성을 높였다. Ascari와 동료들[4]은 기존의 신경망 관련 연구의 실험 대상을 절차 지향 언어인 프로그램에서 객체지향 프로그램에 적용하였다. 기존의 신경망 기반 결함 위치 추정 연구의 실험 대상은 C 절차 지향 언어인 프로그램이었다. 신경망 기반 결함 위치 추정 기법이 실제로 객체 지향 프로그램에서도 적용이 가능한지에 대한 기존의 문제를 해결하기 위해 Ascari와 그의 동료들은 커버리지 데이터를 구문 단위에서 메서드 단위로 변경하면서 객체 지향 프로그램에 적용하였다. 기존의 절차 지향프로그램 실험 대상에서 객체 지향 프로그램 실험 대상으로 도메인을 바꾸어 적용하였다는 것에 의의가 있다. 기존 신경망 결함 위치 추정 기법의 결함 위치 추정 성능 향상 및 다른 언어 프로그램의 적용을 위하여 많은 연구들이 진행되어 왔으며 다양한 방

3. 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법

3.1 신경망 기반 결함 위치 추정

신경망 기반 결함 위치 추정[2]은 테스트 케이스의 실행으로 얻은 커버리지-테스트 결과 쌍을 이용하여 프로그램의 각 구문 결함 의심도를 계산하여 소프트웨어 개발자가 결함을 찾도록 도와준다. 신경망 기반 결함 위치 추정의 아이디어는 실패하는 테스트 케이스들이 자주 커버하는 구문이 결함이 가능성이 높으며, 성공하는 테스트 케이스들이 자주 커버하지 않는 구문이 결함이 가능성이 높다는 원리를 가지고 각 테스트 케이스의 실행으로 얻은 커버리지-테스트 결과를 이용하여 신경망을 학습 후 각 구문 의심도 값을 구한다. 구문 의심도가 큰 것부터 작은 것 순으로 정렬하여, 개발자가 의심도 순으로 검사함으로 결함을 찾을 수 있도록 도움을 준다. 신경망 학습 및 테스트를 위해서 신경망을 학습하는 훈련 데이터와 신경망을 테스트하는 시험 데이터가 필요로 하다. 신경망 결함 위치 추정은 테스트 케이스의 실행으로 얻은 커버리지-테스트 결과 쌍이 학습 데이터로 사용되며, 각 구문을 하나씩 커버하는 가상 테스트 케이스 집합이 시험 데이터로 사용된다.

그림 1 는 신경망 학습을 위한 데이터 정보를 나타낸다. 신경망 학습을 위해서는 2 가지 종류의 데이터가 필요로 하다. 첫 번째로 신경망 학습을 위한 훈련–라벨 데이터 세트와 학습한 신경망에 새로운 실제 데이터를 넣어서 테스트하는 시험 데이터가 필요로 하다. 그림 1 의 첫번째 행의 S1 ~ S9 은 각 구문을 나타내며, 각 Ri 는 테스트의 결과를 나타낸다. S1~S9 의 값은 어떤 테스트 케이스가 관련 구문을 커버하였으면 1 로 나타내고, 커버하지 않았으면 0 으로 나타낸다. Ri 의 값은 0 이면 테스트 케이스의 결과가 Pass 이고, 1 이면 테스트 케이스의 결과가 Fail 이다. 첫 번 째 열의 T1~T7 은 테스트 케이스를 나타내며 7 개 테스트 케이스가 존재한다는 것이다. 신경망 학습을 위하여 각 테스트 케이스의 S1~S9 의 값이 훈련 데이터로 사용되며, 각 테스트 케이스의 Ri 의 값은 학습의 라벨 데이터로 사용된다.

예를 들어, 테스트 케이스 T1 을 보면 커버한 구문은 S1, S2, S3, S4, S6, S9 이며, 커버하지 않은 구문은 S6, S7, S8 이다. (1 1 1 1 0 1 0 0 1) 이 훈련 데이터로 사용이 되고, T1 의 테스트 케이스의 결과 값은 Ri 값, (0)으로 라벨 데이터로 사용된다. 하나의 테스트 케이스의 커버리지-테스트 결과 데이터 쌍이 (1 1 1 1 0 1 0 0 | 0) 신경망 학습에 사용된다.

그림 2 신경망 후 테스트를 위한 시험 데이터 구조

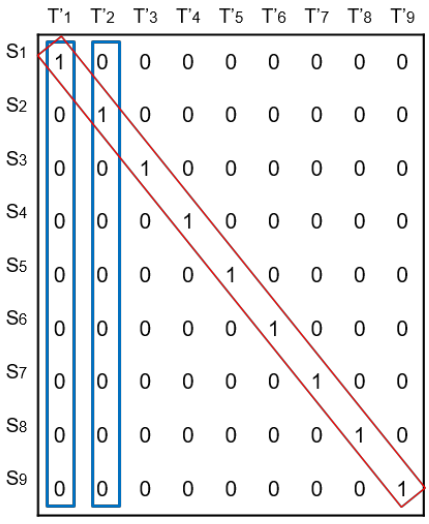


그림 2 는 학습한 신경망에 각 구문을 하나씩 커버하는 테스트 케이스 가상 테스트 케이스인 시험 데이터를 나타낸다. 첫번째 행의 T'1~T'9 은 각각의 테스트 케이스를 나타낸다. 그리고 S1~S9 은 각 구문을 나타낸다. 데이터 구성에서 1 은 구문을 커버한 것을 나타내며, 0 은 구문을 커버하지 않은 것을 나타낸다. 각 테스트 케이스를 살펴보면 하나의 구문을 1 개씩 커버하는 것을 알 수 있다. 각 테스트 케이스의 의미는, 하나의 구문만 커버하고 나머지는 커버하지 않는다는 의미이다.

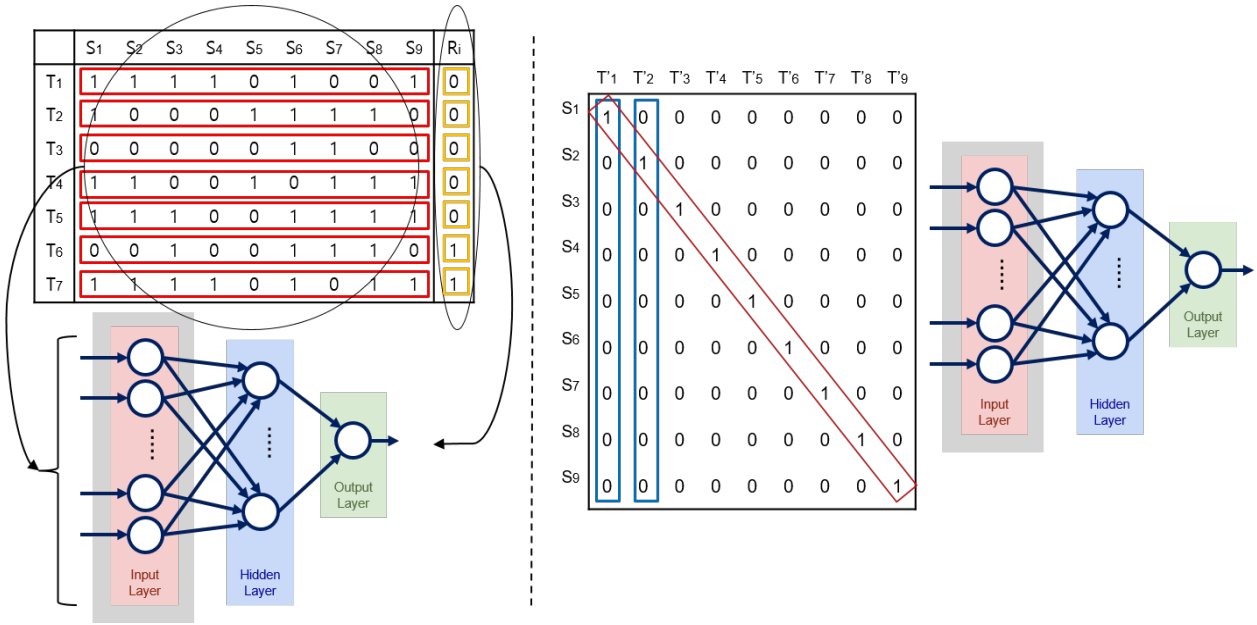
이렇듯, 각 구문의 결함 의심도를 구하기 위하여, 테스트 케이스는 구문의 개수만큼 필요로 하다. 각 구문을 하나씩 커버하는 테스트 케이스를 의미한다. 이러한 테스트 케이스가 실제로 존재할 수 도 있고, 존재하지 않을 수 있지만 그림 2 와 같이 만들게 되면 하나의 구문만 커버하는 의미로서 데이터를 사용할 수 있다.

그림 3 은 테스트 케이스의 커버리지-테스트 결과 데이터를 이용한 신경망 학습 과정과 학습된 신경망에 시험 데이터를 넣어서 의심도 값을 도출해 내는 신경망 결함 위치 추정의 전체 과정이다. 그림 3 의 왼쪽 부분을 보면, 위 그림 1 에서 설명한 훈련 데이터를 신경망의 입력 층과 출력 층에 넣어서 훈련시키는 과정이다. 오른쪽 부분은 훈련된 신경망에 시험 데이터를 입력 층에 넣어서 각 구문의 결함 의심도를 구하는 과정이다. 학습은 각 테스트 케이스의 커버리지-테스트 결과 쌍이 입력과 출력 층에 들어가 신경망이 커버리지-테스트 결과 데이터를 학습한다.

학습된 신경망에 구문을 하나씩 커버하는 커버리지 데이터를 입력 층에 넣어서 각 구문의 결함 의심도를 구한다. 기존 논문 및 관련 확장 논문들을 살펴보면, 결함 위치 추정의 정확성을 높이기 위해서, 신경 망의 노드의 개수를 변경시키는 방향으로 연구를 해왔다. 결함 위치 추정에 있어, 중요한 부분은 신경망의 노드의 개수 및 계층을 늘리는 것 보다, 입력 데이터에 결함과 관련된 의미 있는 정보들을 추가하여 결함 위치 추정의 정확성을 높이는 것이 더 올바른 방향의 방법이다.

3.2 코드 변경 지표

그림 3 신경망 학습 과정 및 학습된 신경망 시험 과정



코드 변경 지표는 코드의 버전 관리 내역을 이용하여, 코드가 언제 작성 및 수정되었는지, 얼마나 빈번하게 수정 되었는지에 대한 코드 변경 지표[5][6]로, 작성 및 수정이 오래된 코드와 덜 빈번하게 수정된 코드는 많은 테스트를 거쳐 코드 결함이 낮은 반면 최근 작성 및 수정된 코드와 빈번하게 변경된 코드는 오래된 코드와 덜 빈번하게 수정된 코드에 비해 테스트를 적게 거쳐 결함이 클 가능성이 높다. 본 연구에 결함 위치 추정의 정확성을 향상시키기 위해, 2 가지 지표인 코드[7]가 언제 작성 및 수정 되었는가에 대한 Age 지표, 코드가 얼마나 빈번하게 수정 되었는가에 대한 Churn 지표를 사용한다. 본 연구에서 코드 버전 관리 내역을 이용하여, 코드가 최근에 언제 작성 및 수정되었는지, 얼마나 빈번하게 수정 되었는지에 대한 정보를 만들어 낸다. Age 의 경우 0~1 사이의 값을 가지게 되며, 0 에 가까울수록 가장 최근에 작성되거나 수정된 코드이며, Churn 도 0~1 사이의 값을 가지게 되며, 1 에 가까울수록 가장 빈번히 수정된 것을 의미한다. 각 코드의 구문과 메서드 단위로 Age 와 Churn 을 버전 관리 내역을 통하여서 구할 수 있다. 코드의 결함에 직접적으로 영향을 미치는 Age, Churn 을 추가 함으로써, 결함 위치 추정의 정확성을 높일 수 있다. 그림 4 은 코드 변경 지표를 기존 신경망 기반 결함 위치 추정에 추가한 훈련 데이터 구조이다. 코드의 버전 관리 내역을

그림 4 기존 신경망 결함 위치 추정 기법에 코드 변경 지표를 추가한 훈련 데이터 구조

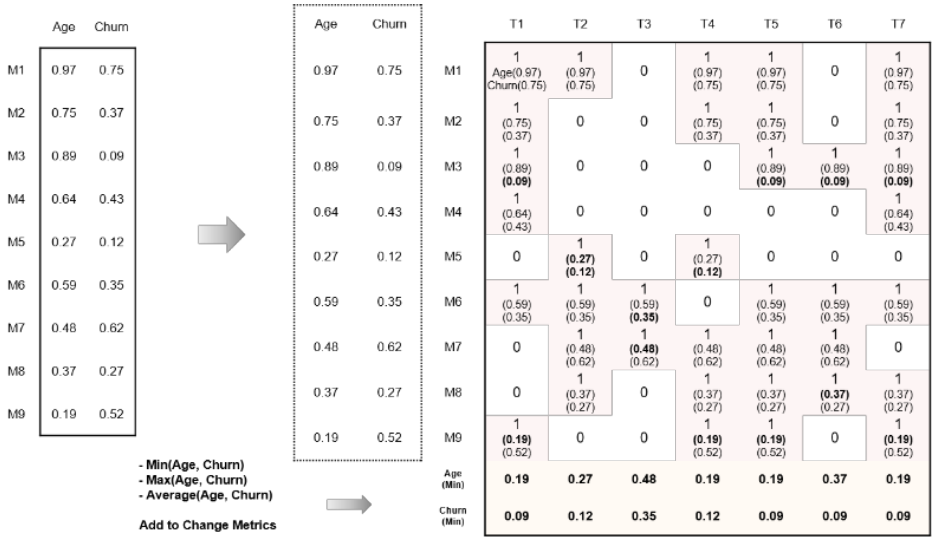


그림 5 코드 변경 지표를 기존 신경망 기반 결함 위치 추정에 추가한 훈련 데이터 구조

	T1	T2	T3	T4	T5	T6	T7
M1	1 (0.97) (0.75)	1 (0.97) (0.75)	0	1 (0.97) (0.75)	1 (0.97) (0.75)	0	1 (0.97) (0.75)
M2	1 (0.75) (0.37)	0	0	1 (0.75) (0.37)	1 (0.75) (0.37)	0	1 (0.75) (0.37)
M3	1 (0.89) **(0.09)**	0	0	0	1 (0.89) **(0.09)**	1 (0.89) **(0.09)**	1 (0.89) **(0.09)**
M4	1 (0.64) (0.43)	0	0	0	0	0	1 (0.64) (0.43)
M5	0	1 **(0.27)** **(0.12)**	0	1 (0.27) **(0.12)**	0	0	0
M6	1 (0.59) (0.35)	1 (0.59) (0.35)	1 (0.59) **(0.35)**	0	1 (0.59) (0.35)	1 (0.59) (0.35)	1 (0.59) (0.35)
M7	0	1 (0.48) (0.62)	1 **(0.48)** (0.62)	1 (0.48) (0.62)	1 (0.48) (0.62)	1 (0.48) (0.62)	0
M8	0	1 (0.37) (0.27)	0	1 (0.37) (0.27)	1 (0.37) (0.27)	1 **(0.37)** (0.27)	1 (0.37) (0.27)
M9	1 **(0.19)** (0.52)	0	0	1 **(0.19)** (0.52)	1 **(0.19)** (0.52)	0	1 **(0.19)** (0.52)
Age (Min)	**0.19**	**0.27**	**0.48**	**0.19**	**0.19**	**0.37**	**0.19**
Churn (Min)	**0.09**	**0.12**	**0.35**	**0.12**	**0.09**	**0.09**	**0.09**

Min(Age, Churn)

	T1	T2	T3	T4	T5	T6	T7
M1	1 **(0.97)** **(0.75)**	1 **(0.97)** **(0.75)**	0	1 **(0.97)** **(0.75)**	1 **(0.97)** **(0.75)**	0	1 **(0.97)** (0.75)
M2	1 (0.75) (0.37)	0	0	1 (0.75) (0.37)	1 (0.75) (0.37)	0	1 (0.75) (0.37)
M3	1 (0.89) (0.09)	0	0	0	1 (0.89) (0.09)	1 **(0.89)** (0.09)	1 (0.89) (0.09)
M4	1 (0.64) (0.43)	0	0	0	0	0	1 (0.64) (0.43)
M5	0	1 (0.27) (0.12)	0	1 (0.27) (0.12)	0	0	0
M6	1 (0.59) (0.35)	1 (0.59) (0.35)	1 **(0.59)** (0.35)	0	1 (0.59) (0.35)	1 (0.59) (0.35)	1 (0.59) (0.35)
M7	0	1 (0.48) (0.62)	1 (0.48) **(0.62)**	1 (0.48) (0.62)	1 (0.48) (0.62)	1 **(0.62)**	0
M8	0	1 (0.37) (0.27)	0	1 (0.37) (0.27)	1 (0.37) (0.27)	1 (0.37) (0.27)	1 (0.37) (0.27)
M9	1 (0.19) (0.52)	0	0	1 (0.19) (0.52)	1 (0.19) (0.52)	0	1 (0.19) (0.52)
Age (Max)	**0.97**	**0.97**	**0.59**	**0.97**	**0.97**	**0.89**	**0.97**
Churn (Max)	**0.75**	**0.75**	**0.62**	**0.75**	**0.75**	**0.62**	**0.75**

Max(Age, Churn)

	T1	T2	T3	T4	T5	T6	T7
M1	1 (0.97) (0.75)	1 (0.97) (0.75)	0	1 (0.97) (0.75)	1 (0.97) (0.75)	0	1 (0.97) (0.75)
M2	1 (0.75) (0.37)	0	0	1 (0.75) (0.37)	1 (0.75) (0.37)	0	1 (0.75) (0.37)
M3	1 (0.89) (0.09)	0	0	0	1 (0.89) (0.09)	1 (0.89) (0.09)	1 (0.89) (0.09)
M4	1 (0.64) (0.43)	0	0	0	0	0	1 (0.64) (0.43)
M5	0	1 (0.27) (0.12)	0	1 (0.27) (0.12)	0	0	0
M6	1 (0.59) (0.35)	1 (0.59) (0.35)	1 (0.59) (0.35)	0	1 (0.59) (0.35)	1 (0.59) (0.35)	1 (0.59) (0.35)
M7	0	1 (0.48) (0.62)	1 (0.48) (0.62)	1 (0.48) (0.62)	1 (0.48) (0.62)	1 (0.48) (0.62)	0
M8	0	1 (0.37) (0.27)	0	1 (0.37) (0.27)	1 (0.37) (0.27)	1 (0.37) (0.27)	1 (0.37) (0.27)
M9	1 (0.19) (0.52)	0	0	1 (0.19) (0.52)	1 (0.19) (0.52)	0	1 (0.19) (0.52)
Age (Average)	**0.6716**	**0.536**	**0.6266**	**0.505**	**0.6057**	**0.5825**	**0.6285**
Churn (Average)	**0.4183**	**0.422**	**0.3916**	**0.4416**	**0.4242**	**0.3325**	**0.3971**

Average(Age, Churn)

이용하여 각 코드의 구문과 메서드에 대한 Age, Churn 값을 구할 수 있다. 실험 대상 프로그램을 메서드 단위로 실험을 진행하였다. 그림 4 의 왼쪽 부분의 표에서 각 메서드들이 가지는 Age 와 Churn 값이 0~1 사이의 값을 가지는 것을 볼 수 있다. 오른쪽 부분은 기존의 신경망 결함 위치 추정 데이터에 코드 변경 지표를 추가한 표이다. 오른쪽 그림의 첫번째 열을 보면 M1~M9 까지의 메서드를 나타내며, 2 개의 행이 추가된 것은 M1~M9 에서 가진 메서드의 Age 와 Churn 에 대한 최소 값을 나타낸다. 실험에서 코드 변경 지표가 신경망 결함 위치 추정 기법에 어떤 영향을 미치는지를 알기 위하여 3 가지 Min, Max, Average 에 대해서 실험을 진행하였으며, 그림 4 은 그림 5 의 Min 에 대한 예시이다. 각 테스트 케이스에 대해 커버한 메서드들의 Age 와 Churn 에 대한 최소 값을 구하여 아래의 2 개의 행인 Age 와 Churn 행에 추가하여 값을 넣게 되며, 이것이 코드 변경 지표를 이용한 신경망 결함 위치 추정에서 훈련 데이터로 사용하게 된다.

그림 5 는 기존 신경망 결함 위치 추정 기법의 훈련 데이터에 테스트 케이스의 최소, 최대, 평균 Age, Churn 코드 변경 지표 값을 추가한 훈련 데이터이다.

그림 6 은 코드 변경 지표를 추가한 시험 데이터 구조를 나타낸다. 기존의 시험 데이터에서, 2 개의 행인 Age, Churn 을 추가하게 되었다. Age, Churn 의 값은 각 테스트 케이스가 하나씩 메서드를 커버하기에, 메서드가 가지는 Age, Churn 값을 넣어서 시험 데이터로 만든다. 시험 과정 시 각 테스트 케이스를 학습된 신경망에 넣으면, 각 메서드에 대한 결함 의심도를 구할 수 있다.

그림 6 코드 변경 지표를 이용한 신경망 결함 위치 추정의 시험 데이터 구조

시험 과정	T'_1	T'_2	T'_3	T'_4	T'_5	T'_6	T'_7	T'_8	T'_9
M_1	1	0	0	0	0	0	0	0	0
M_2	0	1	0	0	0	0	0	0	0
M_3	0	0	1	0	0	0	0	0	0
M_4	0	0	0	1	0	0	0	0	0
M_5	0	0	0	0	1	0	0	0	0
M_6	0	0	0	0	0	1	0	0	0
M_7	0	0	0	0	0	0	1	0	0
M_8	0	0	0	0	0	0	0	1	0
M_9	0	0	0	0	0	0	0	0	1
Age (Actual)	**0.97**	**0.75**	**0.89**	**0.64**	**0.27**	**0.59**	**0.48**	**0.37**	**0.19**
Churn (Actual)	**0.75**	**0.37**	**0.09**	**0.43**	**0.12**	**0.35**	**0.62**	**0.27**	**0.52**

그림 7 는 Age 와 Churn 정보를 추가하여 신경망을 학습하는 과정이다.

그림 7 코드 변경 지표를 이용한 신경망 결함 위치 추정 학습 과정

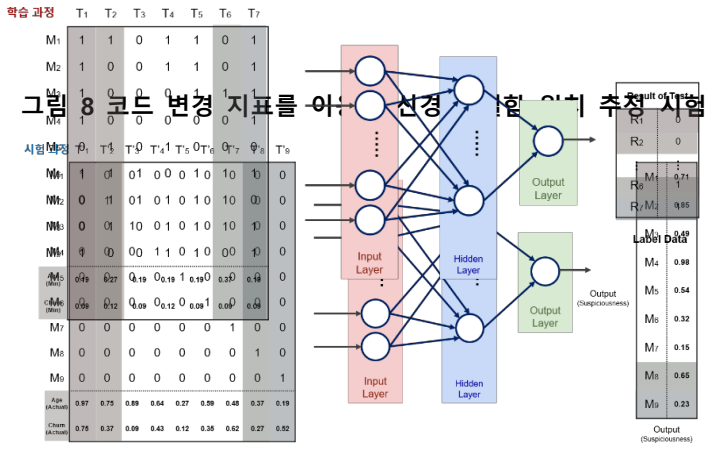


그림 8 코드 변경 지표를 이용한 신경망 결함 위치 추정 시험 과정

입력 층에 들어가는 데이터는 각 테스트 케이스에 대한 메서드 커버 정보와 코드 변경 지표(Age, Churn)값이며, 라벨 데이터는 테스트 결과 데이터로 사용된다.

그림 8 은 코드 변경 지표를 이용한 신경망 결함 위치 추정의 시험 과정을 나타낸다. 그림 8 과 같이 테스트 케이스를 만들면, 하나의 메서드를 커버하는 의미 가지며 시험 데이터로 사용할 수 있다. 각 테스트 케이스 T'1 ~ T'9 까지 학습된 신경망의 입력 층에 넣게 되면 M1 ~ M9 에 대한 결함 의심도 값을 구할 수 있다. 결함 의심도 값은 1 에 가까울수록 결함에 가깝다는 의미이다. 결함 위치 추정 기법을 통해 메서드에 대한 결함 의심도 값이 나온 후 결함 의심도 값이 큰 순서로 나열하여 개발자에게 큰 순서대로 검사 할 수 있게 도와주는 것이 결함 위치 추정 기법의 목표이다. 결함 위치 추정 기법을 평가하기 위한 지표 중 하나로 Exam Score 를 사용한다.

Exam Score 의 공식은 다음과 같다.

$$\text{Exam Score} = \frac{\text{Fault 가 나올 때 까지의 검사 횟수}}{\text{Methods 전체 수}}$$

Exam Score 는 전체 프로그램의 메서드에 대한, 결함 의심도 값이 큰 순서로 나열 한 후 결함이 나올 때 가지 차례대로 검사한 횟수를 나타낸다. Exam Score 가 작을수록 좋은 결함 위치를 잘 찾는 다는 것을 의미한다.

표 1 Exam Score

Methods	Suspiciousness	Rank	Exam Score(%)
M_{2384}	0.98	1	$\frac{1}{5000} * 100 = 0.02\%$
M_{923}	0.85	2	$\frac{2}{5000} * 100 = 0.04\%$
M_{2931}	0.71	3	$\frac{3}{5000} * 100 = 0.06\%$
.	.	.	.
M_{1423}	0.32	4998	$\frac{4998}{5000} * 100 = 99.96\%$
M_{4837}	0.23	4999	$\frac{4999}{5000} * 100 = 99.98\%$
M_{3281}	0.15	5000	$\frac{5000}{5000} * 100 = 100\%$

표 1 은 각 메서드의 Exam Score 을 구한 값을 나타낸 것이다. 표 1 의 M2384 메서드의 결함 의심도 값은 0.98 로 다른 메서드들 보다 가장 높아 검사 순위가 1 순위에 있으며, Exam Score 값은 0.02%으로 가장 낮은 값을 가지게 된다.

4 실험 환경

4.1 연구 질문

본 연구에서 제안한 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법의 정확성 향상 및 확장성을 예측하기 위해서 다음과 같은 연구 질문을 지정하였다.

RQ1. 기존의 신경망 기반 결함 위치 추정과 비교했을 때, 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법은 얼마나 효과적인가? 즉, 코드 변경 지표를 추가하여 실험 했을 때 결함 위치 추정의 정확성에 영향을 미치는지 확인 할 수 있어야 한다. 따라서, 기존 신경망 기반 결함 위치 추정과 코드 변경 지표를 추가한 신경망 기반 결함 위치 추정 기법의 결함 위치 추정하고 그 차이를 비교한다. RQ2. 기존의 실험을 위해 임의로 만든 작은 토이 프로그램에서, 실제 생활에서 사용하고 규모가 큰 프로그램에 적용이 가능한가? 실험 대상이 실제 생활에서 사용하는 규모가 큰 프로그램에도 적용 가능성에 대한 질문이다. 따라서, 기존 작은 토이 프로그램을 실험 대상으로 하는 것에서 벗어나, 실제 생활

에서 사용하는 규모가 큰 프로그램을 실험 대상으로 하여 적용 할 수 있어야 한다. 본 연구는 위의 연구 질문들에 대한 답을 위해 신경망 기반 결함 위치 추정과 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정 기법을 실제 생활에서 사용하는 규모가 큰 Defects4J의 Chart, Lang, Math, Time 4개의 프로그램에 적용하고 해당 기법의 정확성을 측정하였다.

4.2 실험 대상

실험을 위해 사용된 프로그램은 Defeats4J 의 4 개 프로그램을 적용하였다. 4 개 모두 실제 사용자에 의해 사용되는 큰 크기의 프로그램이다. 실험 대상 프로그램들은 사용 용도에 따라 내부 구성이 다르며, 같은 실험 대상 프로그램이더라도 결함 버전에 따라 결함의 종류가 달라 결함 위치 추정을 다양한 환경에서 실험할 수 있다.

표 2 본 연구의 실험 대상 프로그램 정보

Subject Program
(Defects4J : Chart, Math, Time, Lang)

Identifier	Project name	Number of bugs	KLoc	Test KLoc	Test Cases	Dev Years	Fault Type
Chart	JFree Chart	26	96	50	2,205	7	Real Faults
Math	Apache commons-math	106	85	19	3,602	11	
Time	Joda-Time	27	28	53	4,130	11	
Lang	Apache commons-lang	65	22	6	2,245	12	

표 2 는 실험 대상 프로그램에 대한 상세한 정보를 나타낸 것이다. 대상 프로그램은 실제적으로 사용하는 프로그램들이며, JAVA 언어에서 사용되는 프로그램들이다. 본 실험에서는 Dejects4J 에서 제공하는 테스트 케이스를 사용하여, 각 테스트 케이스 별 메서드 커버정보 및 테스트 결과 데이터를 얻게 되었다. 표 2 의 Number of Bugs 는 결함의 버전 개수이며, TestCases 는 테스트케이스의 수를 나타낸다. 최소 실험 대상 프로그램의 코드 크기는 최소 22,000 라인 ~ 최대 96,000 라인으로 되어있다. 테스트 케이스의 수는 2,245~4,130 개로 모두 실제 프로그램이 돌아가는데 테스트하는 테스트 갯수이다.

4.3 실험 구현

본 연구는 코드 변경 지표를 이용한 신경망 기반 결함 위치 추정을 위해 Python 코드로 구현하였다. 본 실험을 위해 Dejects4J 에서 각 테스트 케이스 별 메서드 커버리지 및 결과 데이터 그리고 코드 변경 지표를 얻어낸다 메서드 커버리지 정보는 테스트 케이스에 의해 커버된 메서드이면 1, 그렇지 않으면 0 으로 표현하며, 테스트 결과는 Pass 이면 0 Fail 이면 1 로 표현한다. 코드 변경 지표(Age, Churn)은 0~1 사이의 값을 가지며, 얼마나 수정되거나 작성되지 않고 오랫동안 있는지에 대한 지표인 Age 는 0 에 가까울수록 결함이 높으며, 얼마나 빈번하게 수정되었는지에 지표인 Churn 은 1 에 가까울수록 결함이 높다. 위의 데이터를 이용하여 코드 변경 지표를 추가한 신경망 기반 결함 위치 추정 기법에 적용하여 각 메서드 별로 의심도를 가지게 된다. 메서드의 결함 의심도 값은 0~1 사이의 값으로 정규화하여 나타내며, 1 에 가까울수록 결함일 확률이 높은 순위를 갖는다. 실험은 Ubuntu 16.04 의 운영체제와 8GB 메모리를 갖는 환경에서 진행되었다.

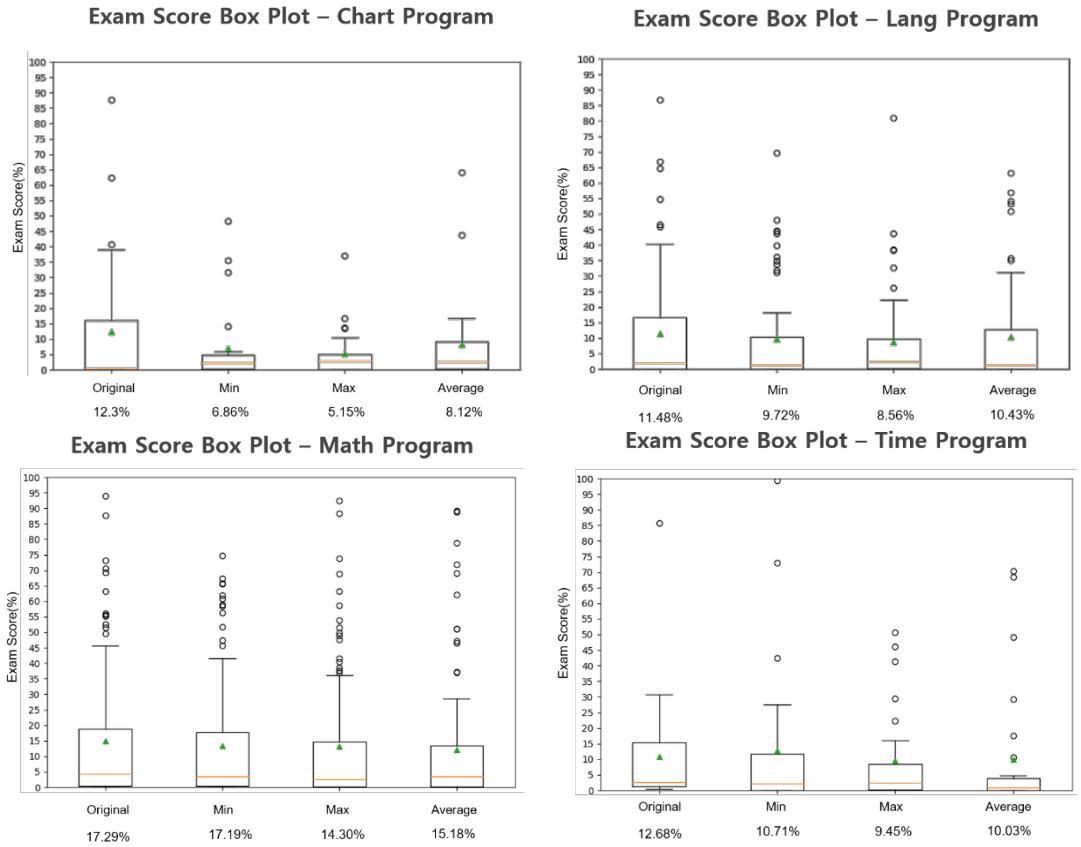
5 실험 결과 및 분석

그림 9 는 실험 대상 프로그램(Dejects4J 의 Chart, Lang, Math, Time)의 실험 결과를 나타낸다.

각 실험 대상 결과의 X 축은 실험의 4 가지 버전을 나타낸다. 차례로 Original 은 코드 변경 지표를 이용하지 않은 기존 신경망 결함 위치 추정 기법을 나타내고, 코드 변경 지표를 3 가지 버전인 Min, Max, Average 을 적용한 신경망 결함 위치 추정 기법을 나타낸다. Y 축은 총 결함 버전에 대한 기법의 Exam Score 을 나타낸다. X 축 버전 아래의 숫자는 모든 결함 버전에 대한 Exam Score 의 평균을 나타낸다. Exam Score 가 낮을수록 결함 위치 추정의 정확성이 높다는 것을 의미한다.

그림 9 의 Chart 실험 대상의 결과를 보면, Original 의 박스 플롯을 보면 기존 신경망 결함 위치 추정 기법은 결함 버전에서 12.3%의 값을 기록하였다. 코드 변경 지표를 적용한 버전인 Min, Max, Average 는 6.86%,

그림 9 대상 프로그램(Defects4J의 Chart, Lang, Math, Time) 실험 결과



5.15%, 8.12%의 값을 기록하였다. 코드 변경 지표를 적용한 버전이 최소 4.18% ~ 최대 7.15% 정확성을 향상시켰다. Lang 실험 대상의 결과도 기존 신경망 결함 위치 추정 기법인 Original 은 11.48%의 값이며, Min, Max, Average 는 9.72%, 8.56%, 10.43%을 기록하였다. 최소 1.05% ~ 최대 2.92%가 정확성을 향상시켰다. Math 실험 대상의 결과는 기존 신경망 결함 위치 추정 기법인 Original 은 17.29%이며 Min, Max, Average 는 각 17.19%, 14.30%, 15.18%을 기록하였다. 최소 0.1% ~ 최대 2.99% 정확성을 향상시켰다. Time 실험 대상의 결과는 기존 신경망 결함 위치 추정 기법인 Original 은 12.68%이며 Min, Max, Average 는 각 10.71%, 9.45%, 10.03%을 기록하여 최소 1.97% ~ 최대 3.23% 정확성을 향상시켰다. 평균 정확성 향상 정도는 2.96%이다. 위 결과와 같이 기존 신경망 결함 위치 추정 기법에 코드 변경 지표를 추가하면 결함 위치 추정의 정확성이 향상되는 것을 알 수 있다.

표 3 기존 관련 연구의 실험 대상 프로그램 정보

Subject Program
(Siemens suite)

Identifier	Project name	Number of bugs	Loc (Lines of code)	Number of executable statements	Test Cases	Fault Types
Siemens suite	Print_tokens	7	565	175	4,130	Manually Seeded Faults
	Print_tokens2	10	510	178	4,115	
	Replace	32	563	216	5,542	
	Schedule	9	412	121	2,650	
	Schedule2	10	307	112	2,710	
	Tcas	41	173	55	1,608	
	Tot-info	23	406	113	1,052	

또한 기존 신경망 결함 위치 추정 연구들은 임의로 만든 작은 토이 프로그램을 실험 대상으로 실험을 진행하였다. 다음 표 3 에 보는 것과 같이 코드의 라인의 수가 최소 173 라인에서 최대 565 라인이며, 결함의 유형도 사용자가 임의로 심어 놓은 결함을 대상으로 하였다. 실험을 위해 만들어진 작은 토이 프로그램을 사용한 것을 알 수 있다. 하지만 4.2 실험 대상의 표 2 인 본 연구에 사용한 실험 대상 프로그램은 최소 22,000 라인에서 최대 96,000 라인으로 규모가 큰 프로그램이며, 실제로 사용되고 있는 오픈소스 JAVA 프로그램으로 결함의 유형도 실제 사용시 나오는 결함으로 구성되어있다. 신경망 결함 위치 추정

기법에서 기존의 구문 단위에서 메서드 단위로 테스트 하여 나온 결과로 신경망을 훈련시켜 위와 같은 실험 결과를 얻게 되었다. 규모가 크고 실제 생활에서 사용하는 프로그램에도 이 기법을 적용할 수 있음을 보였다.

6. 결론 및 향후 연구

본 연구에서는 코드 변경 지표를 추가하여 결함 위치 추정의 정확성을 향상 시킬 수 있는 코드 변경 지표를 이용한 신경망 결함 위치 추정 기법을 제안했다. 코드 변경 지표를 이용한 신경망 결함 위치 추정 기법은 기존 신경망 결함 위치 추정에서 결함에 영향을 미치는 코드 변경 지표를 추가함으로써 신경망 결함 위치 추정 기법의 정확성을 향상시킨다. 기존 작은 토이 프로그램 실험 대상에서 실제 생활에서 사용하는 Dejects4J 의 4 개의 프로그램 Chart, Time, Lang, Math 을 실험 대상에 적용을 하였으며, 코드 변경 지표를 이용한 신경망 결함 위치 추정 기법이 기존 기법보다 높은 정확성을 나타내는 것을 확인 할 수 있었다.

앞으로 커버리지-테스트 결과 데이터 및 코드 변경 지표 이외의 결함과 직접적으로 영향을 미치는 데이터를 추가적으로 고려하여 결함 위치 추정 기법의 정확성을 향상시킬 예정이다.

Acknowledgement

이 논문은 2016 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2016R1C1B1011042)

참고문헌

[1] Wong, W. Eric, et al. A survey on software fault localization. IEEE Transaction Software Engineering, 2016

[2] Wong, W. Eric, and Yu Qi. "BP neural network-based effective fault localization." International Journal of Software Engineering and Knowledge Engineering 19.04 (2009): 573-597.

[3] Zheng, Wei, Desheng Hu, and Jing Wang. "Fault localization analysis based on deep neural network." Mathematical Problems in Engineering 2016 (2016).

[4] L. C. Ascari, L. Y. Araki, A. R. T. Pozo, and S. R. Vergilio, "Exploring machine learning techniques for fault localization,"in Proc. 10th Latin Am. Test Workshop, Buzios, Brazil, Mar. 2009, pp. 1–6.

[5] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction,"
in 2008 ACM/IEEE 30th International Conference on Software Engineering, May 2008, pp. 181–190.

[6] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density,"
in Proceedings of the 27th International Conference on Software Engineering, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 284–292

[7] SOHN, Jeongju; YOO, Shin. FLUCCS: using code and change metrics to improve fault localization.In: Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM, 2017. p. 273-283.

RNN을 이용한 이슈 분류

조 희태°[1], 이 선아[2]
정보과학과, 경상대학교

cht3205@gmail.com[1], saleese@gnu.ac.kr[2]

Issue Classification using RNN

Heetae Cho °[1], Seonah Lee[2]
Department of Informatics, Gyeongsang National University

요 약

소프트웨어를 배포한 이후에도 사용자들이 추가적인 요구사항을 이슈 보고서로 제안한다. 이슈 보고서는 주로 정해진 형식이 없고 다양한 사람들이 작성한다. 문서에 정해진 형식이 없으면 개발자가 문서의 정확한 내용을 파악하기 어렵다. 또한 이렇게 축적한 이슈데이터의 양은 방대하여, 개발자가 이슈데이터를 분석하여 추가 요구사항을 파악하기 어렵다. 본 논문에서는 요구사항을 작성한 이슈 보고서가 소프트웨어의 어느 부분에 관련된 것인지 분류하기 위해 이슈데이터에 Recurrent Neural Network (RNN)을 적용하여 이슈를 자동으로 분류하는 연구를 진행한다. 또한 두 프로젝트를 대상으로 진행한 RNN 분류 모델의 실험 결과, 하나의 프로젝트에서 약 0.56의 정밀도와 0.34의 재현율을 다른 하나의 프로젝트에서는 약 0.67의 정밀도와 0.59의 재현율을 달성하였다. 또한 본 논문에서는 Naïve Bayes, Decision Tree, Random Forest의 머신러닝 방법도 구현하여 RNN 분류 모델과 차이가 없는지 비교하였다. 그 결과 하나의 프로젝트에서는 Decision Tree가 다른 프로젝트에서는 Naïve Bayes 모델이 각각 가장 높은 성능을 보임을 확인하였다. 또한 모든 모델에서 이슈보고서의 제목이 내용보다 더 주요하게 작용함을 확인하였다.

1. 서 론

고객의 요구사항은 소프트웨어 시스템을 개발하는데 필수사항 중 하나이다. 소프트웨어 시스템을 위한 요구사항은 소프트웨어 시스템에 필요한 정보들을 정의한 문서이다. 소프트웨어 요구사항에는 기능적 요구사항과 비기능적 요구사항이 있다. 기능적 요구사항은 완성된 소프트웨어에 포함되어야 할 기능을 명세하며, 비기능적 요구사항은 완성된 소프트웨어가 가져야할 이식성 및 유지보수성 등과 같은 품질과 소프트웨어의 작동 환경과 같은 제약사항을 명세한다.

소프트웨어 계약 및 개발 기간 중의 요구사항은 고객과 개발자가 협의하여 결정한다. 소프트웨어를 출시하면 추가 요구사항은 주로 사용자들이 제안한다. 고객과 개발자가 수립한 요구사항들은 하나의 프로젝트를 위해 일정한 형식의 문서로 기술되기 때문에 해당 요구사항이 무엇을 가리키는지 명확하게 알 수 있다. 이와 대조적으로 소프트웨어 출시 이후 사용자들이 제안하는 요구사항들은 정해진 형식이 없고 내용의 길이에 대한 제약이 없으며, 때때로 해당 요구사항이 무엇을 가리키는지 명시되어 있지 않아 많은 양의 내용을 전부 읽어야 하는 상황도 발생한다. 또한 방대하게 축적되어 갈수록 일일이 읽어내는 것도 쉽지 않다. 이러한 상황은 추가해야 할 요구사항과 해결해야 하는 문제를 개발자가 정확하게 인식할 수 없게 하며, 잘못된 수정에 의한 새로운 문제를 야기할 수 있다.

이러한 문제를 해결하기 위해 기존의 연구들은 요구사항이 가지고 있는 정보들을 통계적으로 분석해 사용자가 작성한 새로운 요구사항을 분류하는 연구를 진행하였다. 예를 들어, 스마트폰의 애플리케이션 사용자 리뷰를 자동으로 기능적 요구사항과 비기능적 요구사항으로 분류하고, 비기능적 요구사항만을 사용성과 신뢰성, 이식성과 성능 총 4가지 범주로 분류하는 연구를 진행하였다[1]. 이 연구는 Bag-of-Words, TF-IDF 등 총 4개의 자연어 처리 기법과 Naïve Bayes, J48, Bagging 총 3개의 기계학습 기법을 모두 조합해 총 12개 모델을 실험하였다[1]. 또 다른 예로, 이슈 추적 시스템들의 변화로 이슈 보고서를 분류하는데 기존의 텍스트 마이닝 기법이 여전히 유효한지를 연구하며, 새로운 마이닝 기법을 제시한 연구도 있었다[2]. 하지만 이 연구들은 요구사항을 소프트웨어가 가진 특정 기능이 아닌 포괄적인 분류 또는 단순히 버그인지 아니면 버그가 아닌지 만을 분류 하였다. 이러한 연구는 문서를 정리하는데 도움을 줄 수는 있지만, 개발자가 해결해야 하는 문제를 인지하거나, 코드의 어떤 부분을 확인해야 하는지 결정하는데 실질적 도움은 크지 않다.

본 논문에서는 정해진 형식이 없는 요구사항을 특정한 항목으로 분류하기 위한 기반 연구로 깃허브의 이슈를 분류하는 연구를 진행한다. 특히 현재 많은 분야에서 높은 성능을 보이고 있는 딥러닝 기법을 적용해 보기로 한다. 이를 위하여 이슈의 제목과 내용을 자연어 처리 기법으로 벡터화 하고, Recurrent Neural Network(RNN) 기법을 사용하여 벡터화된 이슈의 제목과 내용, 레이블

을 학습시켜, 새로운 이슈 보고서의 레이블을 분류하는 모델을 생성하는 방식으로 진행한다. 분류 모델은 제목만 학습하는 모델, 내용만 학습하는 모델, 제목과 내용을 함께 학습하는 모델로 구분하여 학습한다. 또한 기존의 분류 연구에서 뛰어난 성능을 보인 Naïve Bayes, Decision Tree, Random Forest 모델과 비교 실험을 진행한다.

RNN기법을 사용한 우리의 분류 모델은 두 실험 대상 프로젝트에 대해서 각각 0.56의 정밀도와 0.34의 재현율 그리고 0.67의 정밀도와 0.59의 재현율을 달성하였다. 그러나 첫 번째 프로젝트에서는 Decision Tree가 정확도 0.53, 재현도 0.47로 RNN기법을 사용한 모델보다 성능이 높았으며, 두 번째 프로젝트에서는 Naïve Bayes 모델이 정확도 0.81, 재현도 0.70으로 RNN기법을 사용한 모델보다 높았다. 그러나 두 프로젝트를 통해 일관되게 성능이 높은 머신러닝 모델은 없었다. 일관된 사실은 모든 모델을 통하여 제목을 사용한 경우가 내용을 사용한 경우보다 성능이 높았다는 것이다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 본 연구의 실험 모델을 설명한다. 3절에서는 본 연구에서 사용한 자연어 처리 기법을 설명하며, 4절에서는 실험 절차를 설명한다. 5절에서는 실험 결과를 제시하며, 6절에서는 실험 결과에 대한 논의를 진행한다. 7절에서는 관련연구를 기술하며, 8절에서는 결론을 기술한다.

2. 실험 모델

본 2절에서는 실험에서 사용할 RNN 모델에 대해 설명한다. RNN 모델은 연속성이 있는 데이터를 학습시키는데 적합한 딥러닝 모델로, 기본적인 RNN의 동작 구조는 Fig.1과 같으며 아래의 식 (1)과 (2)로 계산된다.

$$s_t = \tanh(W_x x_t + W_s s_{t-1}) \quad (1)$$

$$y_t = W_y s_t \quad (2)$$

식에서의 s_t는 RNN모델의 각 셀의 상태를 나타낸다. W_x, W_s. W_y는 각각 입력, 상태, 출력의 가중치를 나타내며, x_t와 y_t는 t시간의 입력과 출력을 나타낸다.

식 (1)은 상태 s_t의 계산하는 방식을 보여준다. t시간에 RNN모델에 입력 x_t가 들어가게 되면 t-1시간에 계산되었던 상태 s_{t-1}와 상태의 가중치 W_s 를 곱하고 입력 x_t와 입력의 가중치 W_x 를 곱하여 둘의 합과 tanh()함수를 통해 t시간의 상태 s_t를 계산한다. 식 (2)는 t시간까지의 입력에 대한 출력을 계산하는 식으로 t시간의 상태 s_t와 출력의 가중치 W_y를 곱하여 계산한다.

이러한 기본적인 RNN은 깊은 학습에서 문제가 발생할 수 있는데, 이를 해결하기 위해 생겨난 많은 변형

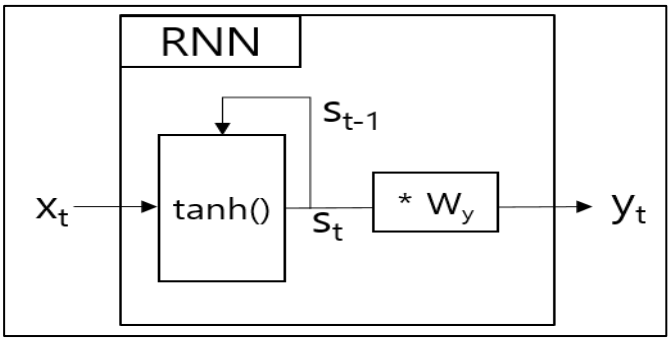


Figure.1 기본적인 RNN의 동작 구조

RNN 중 가장 대표적인 두가지는 long short-term memory (LSTM)[3]와 gated recurrent unit (GRU)[4]이다. 이 두가지 변형은 기본적인 RNN보다 계산이 복잡하지만 깊은 학습에 문제를 줄여준다. 그 중 우리는 GRU를 사용하며 자세한 설명은 4.4절에 있다.

3. 실험 모델을 위한 자연어 처리

본 3절에서는 이슈 보고서의 제목과 내용을 모델에 학습시킬 형태로 바꾸는 자연어 처리 기법에 대해 설명한다.

3.1 Word2Vector

Word2Vector[1]는 Continuous Bag-of-Words(CBOW)와 Skip-gram이라는 두가지 기법이 존재한다. CBOW는 Fig.2과 같이 주변의 단어로 목적 단어를 학습하여 목적 단어를 대표하는 벡터를 생성하는 기법이다. Skip-gram은 Fig.3와 같이 목적 단어로 주변의 단어를 학습하여 벡터를 생성하는 기법이다.

본 논문에서는 보다 정확한 단어의 벡터화를 위해 이미 CBOW 로 학습된 구글에서 제공하는 Word2Vector 모델을 사용한다[2]. 해당 Word2Vector 모델은 구글 뉴스를 기반으로 약 3만개의 영어 단어가 학습되어 있다. 우리는 각 이슈를 제목과 내용으로 나누고, 다시 각각을 단어 단위로 나눈다. 다음 Word2Vector모델을 사용하여 단어의 순서를 유지하며 각 단어를 벡터화 하였다.

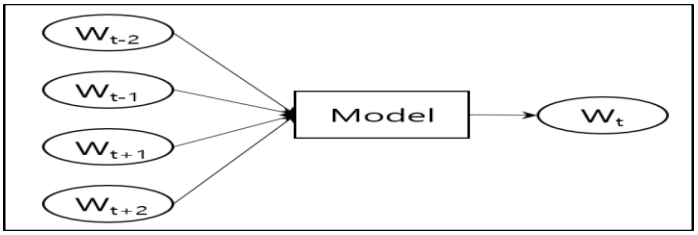


Figure.2 CBOW

[1] https://code.google.com/archive/p/word2vec
[2] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

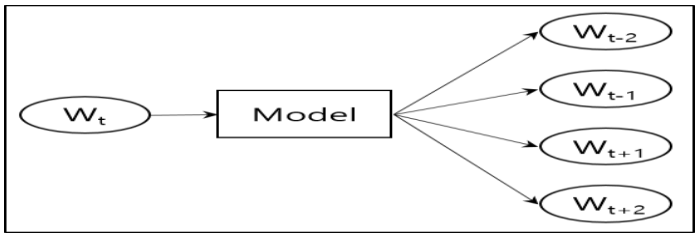


Figure.3 Skip-gram

```
Map = {'i':0, 'you':1, 'he':2, 'she':3,
       'likes':4, 'bird':5, 'dog':6, 'cat':7}
s1 = he likes dog.
s2 = she likes cat.
BoW = [0, 0, 1, 1, 2, 0 ,1, 1]
```

Figure.4 Example of Bag-of-Words

3.2 Bag-of-Words

Bag-of-Words(BoW)는 단어와 해당 단어의 인덱스가 매핑된 테이블을 기반으로 생성되며, BoW를 통한 벡터화의 예시는 Fig.4와 같다.

먼저 매핑 테이블 Map의 전체 길이만큼 0으로 초기화된 리스트 BoW를 생성한다. 다음으로 벡터화 하고자 하는 문장 s1, s2를 단어 단위로 자르고, 각 단어가 매핑 테이블에 존재 한다면, 리스트의 해당 단어의 인덱스 값에 1을 더한다. 즉, BoW는 문장에 등장한 단어의 개수를 세어 벡터화하는, 자연어를 처리하는 간단한 기법 중 하나이다.

해당 기법의 사용을 위해 우리는 먼저 모든 이슈를 읽어 단어의 집합을 만들고 각각의 단어에 고유의 인덱스를 매핑시켰다. 그리고 각 이슈를 제목과 내용으로 나누고 각각을 다시 단어 단위로 나눠 등장한 단어와 등장 횟수를 세어 리스트를 생성해 해당 이슈의 제목과 내용을 벡터화 하였다.

4. 실험 절차

본 4절에서는 우리가 진행한 실험의 절차를 기술한다. 4.1절은 연구 질문을, 4,2절은 실험 대상 프로젝트를 기술한다. 4.3절은 데이터 수집 및 전처리 과정을, 4,4절은 모델 학습을 설명한다. 4.5절과 4.6절은 모델 평가와 비교를 설명한다.

4.1 연구 질문

우리는 이슈를 요구사항별로 분류하기 위한 기반 연구로 이슈를 레이블에 맞춰 분류하는 연구를 진행한다. 본 연구에서 우리는 두개의 연구 질문에 대한 답을 찾기를 기대한다.

RQ.1 간단한 RNN 분류 모델의 성능은 어떠한가?
RQ.2 분류에서 간단한 기계학습 모델과 성능을 비교했을 때 어느 모델이 더 우수한가?

먼저 RQ.1은 모델의 구조를 깊게 생각하지 않고 단순히 GRU 레이어만을 사용한 RNN 모델이 분류에서 얼마나 성능을 낼 수 있는지를 알아보기 위한 연구 질문이며, RQ.2는 단순한 RNN 모델과 동일한 노력이 드는 간단한 기계학습 모델을 구현해 비교함으로써 우리의 모델을 객관적으로 평가할 기반을 마련하는 연구 질문이다.

4.2 실험 대상 프로젝트

우리는 깃허브 오픈소스 저장소에서 실험 대상 프로젝트를 선정하였다. 선정한 기준은 첫째, 불특정 다수가 작성한 이슈데이터가 많은 프로젝트여야 한다. 둘째, 이슈데이터가 대략적인 요구 사항을 표현할 수 있는 레이블로 분류되어 있어야 한다. 마지막으로, 프로젝트의 외부자가 이해하기 쉬운 요구사항이어야 한다. 이는 분석의 용이성을 위해서 필요하다. 이러한 기준에서 통합개발환경의 에디터인 ATOM프로젝트와 VScode, 두 개의 프로젝트를 선정하였다.

먼저 ATOM프로젝트는 전체 이슈가 18,010개로서 분석하기 충분한 양을 가지고 있으며, UI, Network 등 요구사항으로 분류할 수 있는 레이블을 가지고 있었다. 우리는 ATOM프로젝트의 이슈에 태그 될 수 있는 74개의 레이블 중 추가 요구사항과 연관성이 있는 'auto-indent', 'enhancement', 'network', 'proposal', 'typography'와 'ui' 총 6개의 레이블을 선택하였다.

Table.1 Number of Issues

(a) Project Atom

Non-labeled		7547
Label	Auto-indent (Indent)	31
	Enhancement (Enhance)	996
	Network	82
	Proposal	23
	Typography (Typo)	16
	UI	41
	Others	9274
Total		18010

(b) Project VScode

Non-labeled		11152
Label	diff-editor	127
	editor-autoindent (Indent)	246
	editor-find	457
	file-explorer	875
	keybindings	378
	UX	811
	Others	51156
Total		65202

각 레이블에 속하는 이슈 개수는 Table. 1(a)에서 볼

수 있다. 전체 18,010개의 이슈 중 레이블이 붙어 있는 이슈는 10,463개 였으며 그렇지 않은 이슈는 7,547개 였다. 레이블이 붙어 있는 이슈 중 우리의 실험 대상 레이블인 auto-indent는 31개, enhancement는 996개, network는 82개, proposal은 23개, typography는 16개 ui는 41개 였고, 이 외의 이슈들은 대상이 아닌 레이블이 태그되어 있었다.

다음으로 VScode프로젝트는 전체 이슈가 65,202개로서 ATOM프로젝트의 이슈 개수보다 약 세배가 많았다. 해당 프로젝트의 이슈에서는 'diff-editor', 'editor-autoindent', 'editor-find', 'file-explorer', 'keybindings', 'ux'로 총 6개의 레이블을 선택했다. 가급적 ATOM프로젝트와 유사한 레이블을 활용하려고 하였으나, 두 프로젝트간의 레이블의 유사성이 높지 않았다. 유사하게 찾은 것은 autoindent부분과 ux부분이었다.

각 레이블에 속하는 이슈 개수는 Table. 1(b)에서 볼 수 있다. 총 65,202개의 이슈 중 레이블이 붙어있는 이슈는 54,050개 였으며 그렇지 않은 이슈는 11,152개 였다. 레이블이 붙은 이슈 중 실험 대상 레이블인 diff-editor는 127개, editor-autoindent는 246개, editor-find는 457개, file-explorer는 875개, keybindings는 378개, UX는 811개가 있었다.

4.3 데이터 수집 및 전처리

우리는 먼저 깃허브에 있는 통합개발환경을 구현한 프로젝트인 ATOM와 VScode의 모든 이슈 데이터를 크롤링하여 이슈의 번호 순으로 저장하였다.

데이터 전처리 과정에서는 먼저 html로 저장된 이슈들을 파싱하여 제목과 내용, 레이블을 추출하였다. Html을 파싱하여 우리가 원하는 레이블을 가진 이슈들을 찾은 결과는 Table.1과 같다.

다음 각 이슈 별로 알파벳이 아닌 숫자나 특수문자를 포함한 단어를 전부 제거하고, 모든 단어를 소문자로 바꾸었다. 하지만 stop-words의 제거는 기존 연구에서 성능을 떨어뜨릴 수 있다는 의견에 따라 제거하지 않았다[5]. 그후 3.1절에서 설명한 구글의 Word2Vector 모델을 사용해 각 문장을 벡터화 하여 분류 모델에 학습시키기 위한 형태로 바꾸었다.

4.4 모델 학습

본 연구에서 우리가 사용한 모델은 RNN의 변형 레이어인 bidirectional gated recurrent unit 레이어와 선형레이어로 이루어진 단순한 모델로 Fig.5와 같다. Bidirection을 사용한 이유는 기존의 연구에서 단일로 가는 레이어 보다 뛰어난 성능을 보였기 때문이다[6]. 모델 학습에 사용되는 최적화 함수는 Adam[7]을 사용하고 손실 함수는 식 (3)으로 계산되는 CrossEntropy를 사용하였다.

$$\text{CrossEntropy} = -\frac{1}{N}\sum_{n=1}^{N}[y_n log\hat{y} + (1-y_n)\log(1-\hat{y})] \quad (3)$$

식 (3)에서 N은 전체 데이터의 개수이며, y_n은 n번째 출력을 가리키며, $\hat{y}$은 실제 데이터를 가리킨다. y_n이 $\hat{y}$과 비슷하다면 작은 수를 많은 차이가 있다면 큰 수를 반환하여 최종적으로 최소화하는 쪽으로 학습하게 된다.

학습에는 하나의 이슈에 대해 총 3가지 형태의 데이터를 사용하였다. 첫번째는 이슈의 제목만을 가지고 학습시킨 모델이며, 두번째는 이슈의 내용만을 가지고 학습시킨 모델이고, 마지막은 제목과 내용을 함께 학습시킨 모델이다.

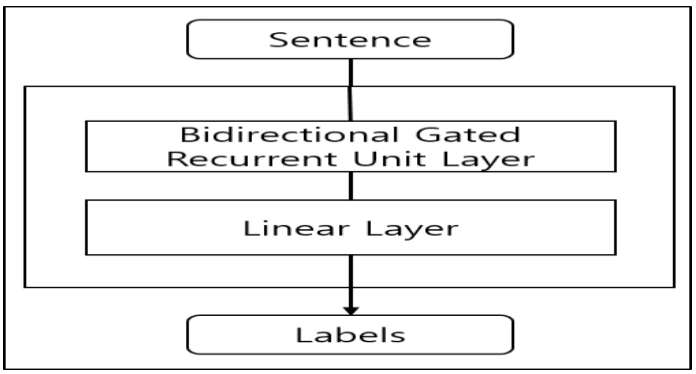


Figure.5 Model Architecture

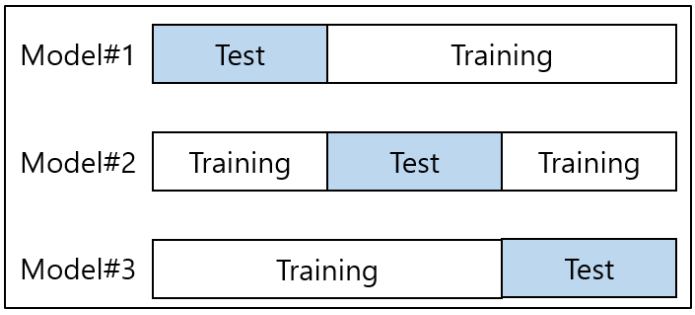


Figure.6 3-fold cross validation

4.5 모델 평가

테스트 모델에서는 추천의 정확도를 올리기 위해 추천된 전체 레이블 값의 합이 1이 되도록 소프트맥스 함수를 이용해 모델이 추천한 값이 정답일 가능성이 0.9를 넘을 때만 추천하도록 하였다.

테스트 방식은 3겹 교차검증 방법을 사용하였다. 우리가 실시한 3겹 교차검증의 방법은 Fig.6과 같이, 작성날짜로 정렬한 데이타를 1/3씩 나누어 평가하는 방식이다. 평가의 첫번째 모델은 전체 데이터의 처음 1/3을 평가에 사용하며 나머지 2/3을 학습에 사용한다. 두번째 모델은 처음과 마지막에서 각각 1/3씩을 학습에 사용하고 중간의 1/3을 평가에 사용하며, 마지막 세번째 모델은 데이터의 처음 2/3을 학습에 마지막 1/3을 평가에 사용한다. 교차검증 방법은 데이터의 내용이 한쪽으로 치우침을 방지 하여, 모델의 평균적인 성능을 측정

할 수 있다는 장점이 있다.

3겹 교차검증을 이용한 평가 결과는 각 학습 데이터별로 생성된 3개 모델의 척도별 평균으로 보인다. 척도는 4.7을 참조한다.

4.6 모델 비교

우리는 RNN모델을 사용한 분류 모델이 보이는 성능의 객관적인 비교를 위해 기존의 분류 연구에서 우수한 성능을 보인 기계학습 모델과 성능을 비교한다. 먼저 우리가 적용한 기계학습 모델과 설정 정보는 Table.2와 같다. Naïve Bayes 모델은 멀티노미얼 방식을 사용하였다. Decision Tree 모델은 criterion을 gini로 설정하다. Random forest 모델 역시 criterion은 gini로 설정하였고 트리의 개수는 10으로 설정하였다. 이 모델들은 단어의 등장확률이나 단어의 유무로 학습 및 판단 하는 모델들로 이슈의 내용을 단어별로 벡터화 하는 방법보다 전체를 BoW를 사용하는 것이 더욱 뛰어난 성능을 보일것이라 생각하여 세가지 모델 모두 자연어 처리 기법으로 BoW를 사용하였다. 또한 이 세 모델도 동일한 데이터를 이용해 3겹 교차검증을 진행하였다.

Table.2 Machine Learning Models

Model	Setting
Naïve Bayes	Multinomial Naïve Bayes, BoW
Decision Tree	criterion='gini', BoW
Random Forest	criterion='gini', n_estimators=10, BoW

4.7 평가 척도

본 논문의 평가는 레이블 추천 결과에 대하여 일반적으로 사용되는 추천율(Recom), 정밀도(Pr)와 재현율(Rc), F1 score 사용하였으며 각각 식 (4), (5), (6), (7)으로 계산하였다

$$\text{추천율} = \frac{\text{레이블 추천이 발생한 횟수}}{\text{레이블 추천을 시도한 횟수}} \quad (4)$$

$$\text{정밀도}_i = \frac{\text{추천한 } Label_i\text{가 정답인 횟수}}{\text{모델이 } Label_i\text{라고 추천한 횟수}} \quad (5)$$

$$\text{재현율}_i = \frac{\text{추천한 } Label_i\text{가 정답인 횟수}}{\text{전체 } Label_i\text{의 개수}} \quad (6)$$

$$F1\ score = 2 * \frac{avg(\text{정밀도}) * avg(\text{재현율})}{avg(\text{정밀도}) + avg(\text{재현율})} \quad (7)$$

먼저 추천율은 레이블 추천을 시도한 횟수 대비 입력에 대해 모델이 추천을 발생 시킨 비율이다. 추천율이 높다는 것은 각 이슈를 레이블로 분류할 수 있다는 의미이고, 추천율이 낮다는 것은 각 이슈를 레이블로 분류할 수 없다는 의미이다.

다음으로 정밀도와 재현율은 각 레이블 별로 계산하였으며, 정밀도는 모델이 Label i라고 추천한 횟수 대비 해당 입력의 실제 정답이 레이블 i인 비율로 우리의 모델이 얼마나 정확하게 분류할 수 있는가를 나타낸다.

재현율은 테스트 데이터에 포함된 레이블 i의 전체 개수 대비 모델이 추천한 레이블 i가 정답인 비율로 우리의 모델이 실제 데이터에서 레이블을 얼마나 포괄하여 분류할 수 있는 지를 나타낸다.

마지막으로 정밀도와 재현율 사이에는 반비례관계가 존재하기 때문에, F1 score는 이 둘을 이용한 조화평균으로 분류 모델의 실제 성능을 평가하는 지표가 된다.

5. 실험 결과

본 5절에서는 우리의 모델을 사용해 이슈를 분류한 실험 결과를 보인다. 또한 우리는 분류 모델의 비교를 위해 분류에 자주 사용되는 Naïve bayes, Decision Tree, Random Forest모델을 구현하여 동일한 데이터로 평가하였다. 이러한 실험 결과를 바탕으로 4.1절의 연구 질문에 대한 답을 논의한다.

5.1 RQ.1 간단한 RNN 분류 모델의 성능은 어떠한가?

실험의 결과는 Table.3과 같다. 먼저 ATOM 프로젝트의 실험 결과인 Table3(a)를 보면 제목만 사용하였을 때 정밀도 0.46, 재현율 0.32이고, 내용만 사용하였을 때 정밀도 0.37, 재현율 0.28이었으며, 마지막으로 제목과 내용을 모두 사용하였을 때 정밀도 0.56, 재현율 0.34이었다. 실험의 결과 가장 성능이 좋은 모델은 제목과 내용을 함께 사용한 모델로 F1 score는 0.42 였다. 추천율은 세개의 모델에서 전부 95%를 넘어 큰 차이가 없었다.

가장 성능 좋았던 제목과 내용을 함께 사용한 모델의 레이블별 결과를 보면, auto-indent는 정확도 0.72, 재현율 0.20, enhancement는 정밀도 0.90, 재현율 0.89, network는 정밀도 0.61, 재현율 0.56으로써 전체적으로 정밀도와 재현율이 가장 높은 레이블은 enhancement이다. 이러한 결과는 Table.1(a)에서 enhancement 레이블이 다른 레이블에 비해 그 수가 월등히 많아 더 많은 학습 데이터가 포함 되었기 때문이다. 반면 proposal 레이블 같은 경우 하나도 맞추지 못했다. 분석을 진행한 결과 우리의 RNN 모델은 proposal 이슈의 입력에 대해 99%를 enhancement라고 출력하였다. 이는 enhancement가 proposal에 비해 더 많은 양의 데이터를 가지고 있기 때문인 것도 있지만, proposal이 enhancement의 부분집합일수도 있기 때문이다.

다음으로 VScode의 실험결과인 Table.3(b)를 보면 제목만 사용하였을 때 정밀도 0.67, 재현율 0.59이고, 내용만 사용하였을 때 정밀도 0.46, 재현율 0.27이었으며, 마지막으로 제목과 내용을 모두 사용하였을 때 정밀도 0.56, 재현율 0.38이었다. 내용만 학습시킨 모델은 F1 score 0.31이고 제목과 내용을 함께 학습시킨 모델은 0.43으로써 ATOM프로젝트 실험결과의 성능면에서는 큰 차이를 보이지 않았다. 반면 제목만을 사용해

Table.3 Results of RNN 3-fold cross validation

(a) Project ATOM

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
Indent	0.72	0.30	0.13	0.07	0.72	0.20
Enhance	0.88	0.96	0.88	0.91	0.90	0.89
Network	0.70	0.45	0.57	0.44	0.61	0.56
Proposal	0	0	0	0	0	0
Typo	0.22	0.13	0.19	0.13	0.76	0.31
UI	0.23	0.07	0.41	0.15	0.35	0.05
Avg	**0.46**	**0.32**	**0.37**	**0.28**	**0.56**	**0.34**
Recom	**0.96**		**0.95**		**0.95**	
F1 score	**0.38**		**0.32**		**0.42**	

(b) Project VScode

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
diff-editor	0.69	0.50	0.30	0.08	0.56	0.34
Indent	0.70	0.62	0.58	0.27	0.56	0.29
editor-find	0.66	0.58	0.45	0.16	0.73	0.33
file explorer	0.75	0.74	0.65	0.25	0.63	0.37
keybindings	0.57	0.52	0.37	0.19	0.39	0.24
UX	0.63	0.60	0.41	0.67	0.48	0.72
Avg	**0.67**	**0.59**	**0.46**	**0.27**	**0.56**	**0.38**
Recom	**0.92**		**0.73**		**0.85**	
F1 score	**0.63**		**0.31**		**0.43**	

Table.4 Performance of Machine Learning Models

(a) Evaluation results of Naïve Bayes for ATOM

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
Indent	0.93	0.30	0.17	0.03	0.64	0.33
Enhance	0.88	0.97	0.89	0.97	0.91	0.94
Network	0.67	0.55	0.77	0.72	0.72	0.83
Proposal	0	0	0.40	0.10	0.14	0.10
Typo	0.00	0.00	0.00	0.00	0.17	0.06
UI	0.13	0.05	0.47	0.12	0.45	0.27
Avg	**0.44**	**0.31**	**0.45**	**0.32**	**0.51**	**0.42**
F1 score	**0.36**		**0.38**		**0.46**	

(b) Evaluation results of Decision Tree for ATOM

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
Indent	0.35	0.33	0.29	0.23	0.51	0.37
Enhance	0.89	0.91	0.89	0.92	0.90	0.93
Network	0.66	0.60	0.71	0.61	0.73	0.67
Proposal	0	0	0.43	0.33	0.34	0.32
Typo	0.40	0.38	0.40	0.34	0.40	0.36
UI	0.17	0.15	0.21	0.12	0.30	0.20
Avg	**0.41**	**0.39**	**0.49**	**0.43**	**0.53**	**0.47**
F1 score	**0.40**		**0.46**		**0.50**	

(c) Evaluation results of Random Forest for ATOM

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
Indent	0.52	0.20	1.00	0.10	0.50	0.20
Enhance	0.87	0.99	0.86	1.00	0.87	0.99
Network	0.85	0.46	0.84	0.34	0.90	0.50
Proposal	0	0	1.00	0.18	0	0
Typo	0.17	0.07	0.00	0.00	0	0
UI	0.33	0.02	0.33	0.02	0	0
Avg	**0.52**	**0.20**	**1.00**	**0.10**	**0.50**	**0.20**
F1 score	**0.35**		**0.39**		**0.32**	

(d) Evaluation results of Naïve Bayes for VScode

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
diff-editor	0.88	0.22	0.33	0.01	0.88	0.21
Indent	0.92	0.74	0.88	0.49	0.91	0.81
editor-find	0.76	0.71	0.64	0.56	0.72	0.79
file explorer	0.71	0.89	0.57	0.82	0.80	0.88
keybindings	0.82	0.63	0.77	0.44	0.82	0.69
UX	0.71	0.74	0.67	0.71	0.75	0.79
Avg	**0.80**	**0.66**	**0.64**	**0.50**	**0.81**	**0.70**
F1 score	**0.72**		**0.57**		**0.75**	

(e) Evaluation results of Decision Tree for VScode

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
diff-editor	0.68	0.58	0.39	0.27	0.71	0.61
Indent	0.69	0.66	0.51	0.39	0.68	0.67
editor-find	0.66	0.59	0.43	0.39	0.59	0.56
file explorer	0.75	0.73	0.55	0.56	0.74	0.74
keybindings	0.72	0.67	0.56	0.45	0.69	0.65
UX	0.57	0.65	0.47	0.57	0.58	0.61
Avg	**0.68**	**0.65**	**0.49**	**0.44**	**0.67**	**0.64**
F1 score	**0.66**		**0.46**		**0.65**	

(f) Evaluation results of Random Forest for VScode

	Title		Context		Title+Context	
	Pr	Rc	Pr	Rc	Pr	Rc
diff-editor	0.76	0.57	0.57	0.23	0.89	0.60
Indent	0.76	0.66	0.57	0.36	0.77	0.74
editor-find	0.68	0.66	0.47	0.44	0.63	0.64
file explorer	0.73	0.79	0.53	0.65	0.74	0.82
keybindings	0.71	0.61	0.58	0.31	0.77	0.65
UX	0.62	0.65	0.51	0.57	0.65	0.65
Avg	**0.71**	**0.66**	**0.54**	**0.43**	**0.74**	**0.68**
F1 score	**0.68**		**0.48**		**0.71**	

학습시킨 모델은 F1 score값이 0.63으로 가장 우수한 성능을 보였다. VScode를 사용한 실험의 결과는 제목만을 학습 시킨 모델과 다른 모델 사이에 성능과 추천율에서 많은 차이를 보였다.

가장 성능 좋았던 제목을 사용한 모델의 레이블별 결과를 보면, indent는 정확도 0.70, 재현도 0.62, file explorer는 정확도 0.75, 재현도 0.74 등으로 전체적으로 정밀도와 재현율이 가장 높은 레이블은 file explorer이다. Table.1(b)에서 file explorer 레이블 또한 그 수가 가장 많았다. 그러나 ATOM의 결과처럼 하나도 레이블을 맞추지 못하는 경우는 없었다. 이는 ATOM프로젝트보다 데이터의 치우침이 적으며 각각의 데이터 양 또한 많기 때문일것으로 추정한다.

5.2 RQ.2 분류에서 간단한 기계학습 모델과 성능을 비교했을 때 어느 모델이 더 우수한가?

기계학습 모델의 평가결과는 Table.4와 같다. Table.4의 (a), (b), (c)는 각각 ATOM프로젝트를 대상으로 진행한 Naïve Bayes, Decision Tree, Random Forest모델의 결과를 나타낸다. 예를 들어 Naïve Bayes 모델의 경우 제목만 사용한 경우 F1 score가 0.36, 내용을 활용한 경우 0.38, 그리고 제목과 내용을 함께 사용한 경우 0.46이었다. 이와 유사하게 Decision Tree도 제목과 내용을 함께 학습시켰을 때 F1 score가 0.50으로 또한 정밀도 정밀도 0.53과 0.47로 제목만 혹은 내용만 학습한 모델보다 성능이 좋았다.

반면, Random Forest모델의 경우, 제목과 내용을 같이 학습 시켰을 때가 F1 score 0.32로 가장 낮았다. 제목만 학습시켰을 때 정밀도 0.52, 재현도 0.20이고 내용만 학습시켰을 때 정밀도 1.00과 재현율 0.10으로 가장 좋은 성능을 보였다. 그런데 Random Forest 모델의 경우에서 3가지 레이블은 전혀 정답을 추천받지 못하는 결과를 보였다. 이는 우리가 설정한 10개의 트리 중 정답 레이블이 아닌 다른 레이블을 추천한 트리가 더 많아 나타난 결과이다.

세 모델 중 가장 성능이 좋은 모델은 Decision Tree모델로 F1 score 값이 0.50이다. 이는 우리의 모델이 보인 F1 score 값 0.42보다 8% 정도 우수한 결과이다. 그러나 정밀도는 우리의 모델이 0.56으로 Decision Tree의 정밀도 0.53보다 높다. 단 재현율이 0.34로 Decision Tree의 재현율 0.47보다 낮아서 위와 같은 결과가 나왔다. 기타, 세 모델 전부에서 enhancement가 가장 높은 정밀도와 재현율을 보였으며 그 이유는 데이터의 양이 가장 많기 때문으로 추정한다. 일반적인 성능 또한 학습된 데이터의 양에 비례하였다.

다음으로 Table.4의 (d), (e), (f)는 각각 VScode프로젝트를 대상으로 진행한 Naïve Bayes, Decision Tree, Random Forest모델의 실험결과이다. 가장 수치가 높은 경우를 각각 살펴보면 Naïve Bayes 모델에서 제목과 내용을 함께 학습한 경우의 F1 score값이 0.75로 제목만(0.72) 혹은 내용(0.57)만 학습한 경우보다 성능이 좋았다. Random Forest모델 또한 제목과 내용을 함께 학습한 경우에서 F1 score 0.71로 제목(0.68) 혹은 내용(0.48)만을 학습한 경우보다 성능이 좋았다. 반면, Decision Tree 모델은 제목만 학습한 경우가 F1 score 0.68로 제목과 내용을 함께 학습한 경우보다 성능이 좋았다.

VScode프로젝트를 대상으로 가장 성능이 좋은 모델은 Naïve Bayes 모델로 F1 score값이 0.75로 평균 0.81의 정밀도와 0.70의 재현율을 달성했다. 이 모델의 성능은 우리 RNN의 모델의 성능인 F1 score값 0.63, 정밀도 0.67, 재현율 0.59를 모두 능가하였다.

6. 논의

본 6절에서는 실험의 과정과 결과에서 발견하고 나타난 가설들에 대해 논의를 진행한다.

6.1 RNN모델의 파라미터 튜닝

본 실험은 windows10 64bit, RAM 16.0G, NVIDIA GeForce GTX 1060 6GB환경에서 진행하였으며, 최종적으로 사용된 RNN모델의 파라미터는 Table.5와 같다. Table.5에서 모델의 입력 크기(Input_size)는 Word2Vector에서 생성되는 단어의 벡터 크기와 같다.

Table.5 Parameters of RNN model

Parameter	Value
Input_size	300
Hidden_size	150
Output_size	6
Num_layers	1
epochs	1000

히든 유닛의 수(Hidden_size)는 해당 레이어에서 학습하는 뉴런의 수로 생각할 수 있으며, 모델의 출력 크기(Output_size)는 우리가 분류하고자 하는 범주의 개수와 같다. 모델의 레이어 수(Num_layers)는 하나이며 총 1000번의 학습을 진행하고, 드롭아웃은 하지 않았다.

우리는 이 모델을 결정하기전에 최적의 모델을 찾기 위해 여러 번의 파라미터 튜닝을 시도했다. 모델의 입력은 변경하지 않았지만, Hidden_size와 레이어의 수, 학습의 횟수를 다양하게 시도해 보았다. 먼저 Hidden_size를 1000까지 50단위로 증가 시켜보았다. 150까지 증가 시킬 때는 성능의 변화를 체감할 수 있었지만 그 이후에는 학습에 시간이 걸릴 뿐 커다란 변화는 보이지 않았다.

다음으로는 레이어의 수를 12까지 증가시켜 보았다. 하지만 데이터가 적어서 그런지 레이어를 늘리고 학습 횟수 200정도에서는 학습이 제대로 이루어 지지 않았다. 학습에는 약 25000초가 걸렸다. 즉 학습이 이루어 지게 하려면 학습의 횟수를 증가시켜야 하지만 단순한 계산

으로 학습 횟수가 1000이 된다면 약 125000초가 걸리게 되는데 이는 34시간에 육박한다. 이렇게 되면 여러 번의 실험 진행이 불가하여 모델의 학습에 걸리는 시간적 효율을 생각해 레이어의 수는 1로 설정했다.

마지막으로 학습 횟수를 1000까지는 250단위로 이후에는 1000단위로 10000까지 늘려보았다. 1000이하의 학습 횟수에서는 점점 성능이 증가했지만 이후에는 별다른 변화가 없거나 떨어지기 까지 했다. 또 모델에 다른 파라미터를 가지는 GRU 레이어를 추가도 해보았지만 성능의 증가는 볼 수 없었다. 따라서 우리의 모델은 최고나 최적이라고 할 수 없지만, 우리가 가진 데이터에서는 최소한 어느 정도의 결과를 낼 수 있는 모델이다.

차후 더욱 성능을 올리기 위해 기존의 연구에서 실시한 좀 더 다양한 모델을 사용해볼 것이며, 시간이 걸리더라도 더욱 깊은 학습을 진행할 예정이다.

6.2 학습 데이터에 따른 결과

본 논문에서는 하나의 이슈에 대해 총 세 가지 형태의 데이터로 변형시켜 RNN 모델을 활용한 이슈 분류 실험을 진행하였다. 그 결과 ATOM 프로젝트에서는 제목과 내용을 함께 사용한 경우가 VScode 프로젝트에서는 제목만을 사용한 경우가 가장 성능이 좋았다. 하나의 경우가 최선의 성능을 보여주지 않기 때문에 가장 좋은 성능의 모델에 대한 선택은 유보한다.

이보다 공통적인 결과는 두 실험 모두 제목만을 학습한 모델이 내용만을 학습한 모델보다 성능이 우수하다는 결과이다. 일반적으로 생각하면 제목은 내용의 핵심만을 요약한 정보라고 할 수 있다. 여기서 우리는 다음과 같은 가설을 세웠다. 이슈의 내용을 잘 요약하면 제목과 내용을 함께 학습 시킬 때의 데이터보다 훨씬 작은 데이터로 더욱 좋은 성능을 보일 수 있을 것이다. 따라서 우리는 가설의 검증을 위해 향후 이슈의 내용을 요약하는 연구와 요약된 이슈를 이용해 분류하는 연구를 진행할 것이다.

6.3 학습 모델에 따른 결과

ATOM프로젝트에서는 Decision Tree모델이, VScode프로젝트에서는 Naïve Bayes모델이 가장 우수한 성능을 보였으며, 우리의 모델보다도 우수한 성능을 보인다. 이러한 결과는 데이터의 양이 방대해 지지 않는 한 다른 기계학습 기반 분류 기법이 딥러닝 기법인 Neural Network 기법보다 더 뛰어날 수 있다는 가설을 세울 수 있다. 또한 동일한 노력으로 결과를 내고자 할 때 쉽게 구현할 수 있는 기계학습 모델이 더 좋은 성능을 보일 수 있다는 가설도 세울 수 있다. 즉, 데이터의 양이 많다면 Neural Network 기법이 더 뛰어나다는 것을 확인하기 위하여, 차후 우리는 더욱 많은 데이터를 수집하고, 새로운 모델 구조에 관한 연구를 진행할 것이다.

7. 관련연구

관련 연구로는 이슈보고서를 분류한 연구와 사용자 리뷰들을 분류한 연구가 있다.

먼저 우리와 같이 깃허브에 존재하는 이슈를 대상으로 분류를 진행한 연구가 있었다[2]. 해당 연구는 기존의 텍스트 정보만을 사용하는 분류 기법과 저자들이 제안한 two-stage기법을 비교하는 연구를 진행하였다. 실험은 SVM, naïve bayes등 네 가지 모델과 기존의 자연어 처리 기법으로 tf-idf를 사용하였으며, 저자들은 텍스트 의미의 perplexity를 추가적으로 사용해 분류를 진행하였으며, 결과 two-stage기법과 SVM 모델을 사용한 분류가 가장 뛰어난 성능을 보였다[2]. 해당 연구는 새로운 분류 기법을 선보였다. 하지만 분류 방식이 이슈가 버그인가 아닌가의 이진 분류인 점이 우리의 연구와는 다르다. 우리 연구는 이슈를 요구사항에 직접 연관시키기 위한 기반 연구로 이슈 레이블에 따라 이슈를 분류하는 연구를 수행하였다.

다음으로 스마트폰의 앱스토어에 작성되는 특정 애플리케이션에 대한 사용자의 리뷰를 대상으로 분류를 진행한 연구가 있었다[1][8]. 그 중 한 연구는 개발자가 유용하게 이용할 수 있는 정보를 사용자 리뷰가 포함하고 있다지만, 그 양이 방대하다는 문제점을 들며 다양한 자연어 처리 기법과 분류 모델을 조합해 사용자 리뷰를 분류하였다. 그 중 가장 뛰어난 결과는 보인 조합은 해당 논문의 저자들이 제안한 AUR-BoW라는 자연어 처리 기법과 Bagging모델의 조합으로 71.4%의 정밀도와 72.3%의 재현율을 보였다. 하지만 해당 논문에서 진행한 리뷰의 분류는 기본적으로 비기능적 요구사항과 기능적 요구사항으로 분류하였으며, 그 중 비기능적 요구사항을 특정한 범주로 나누어 분류하였다[1].

사용자 리뷰를 분류한 다른 연구는 사용자 리뷰가 애플리케이션의 진화에 유용한 정보를 포함하고 있지만 개발자가 직접 분석하기에는 그 양이 너무 많다는 문제점을 해결하기 위해 앙상블 방법을 사용한 모델로 새로운 분류 체계를 제안하여 74%의 정밀도와 59%의 재현율을 보였다. 하지만 해당 연구는 사용자 리뷰를 애플리케이션의 특정 기능별이 아닌 통합적으로 버그나 사용자의 요구 등 집합적인 범주로 분류하였다[8].

또 다른 연구로는 사용자 리뷰를 자연어 처리와 텍스트 분석, 감정 분석 총 세가지 기법을 사용해 분류하는 연구를 진행하였다. 해당 논문은 SVM, J48등 5가지 분류 모델을 사용하며 자연어 처리 기법만 사용한 데이터, 텍스트 분석만 사용한 데이터, 자연어 처리와 감정 분석을 사용한 데이터, 자연어 처리와 텍스트 분석만 사용한 데이터, 세가지 모두를 사용한 데이터로 총 다섯 가지의 데이터 형태를 실험하였으며, 그 중 J48 분류 모델과 자연어 처리, 텍스트 분석, 감정 분석 모두를 사용한 데이터 형태가 75%의 정밀도와 74%의 재현율을 달성해 가장 성능이 좋았다. 하지만 해당 논문 또한 애플리케이션의 특정 기능이 아닌 추가 기능의 요구나 문제 발견 등 집합적인 범주로 분류하였다[9].

다음으로는 요구사항 명세서에서 소프트웨어 개발에 필수적인 보안 요구사항을 명확하게 도출하기 어렵다는 점을 문제점으로 들며, 텍스트 마이닝 기술과 J48 분류 모델을 사용해 보안과 관련된 요구사항 명세서를 인증 권한 부여, 접근 제한 등 특정한 기능으로 분류하는 연구를 진행하였다[10]. 해당 연구는 우리가 목표로 하고 있는 특정 기능으로의 분류와 비슷하지만 요구사항의 범위를 보안 요구사항을 한정하였다.

본 연구는 기존 연구와 목표와 분류 기법에서 차이를 보인다. 목표는 특정 요구사항과 이슈보고서를 연결할 수 있도록 이슈 별 분류를 진행하였으며, 최근 높은 성능을 보인다고 알려진 딥러닝 기법 중 하나인 RNN을 포함하여 요구사항 자동 분류 실험을 진행하였다.

8. 결 론

잘 쓰여진 요구사항은 소프트웨어 개발 및 유지보수의 비용을 줄일 수 있다. 하지만 소프트웨어 유지 및 보수 기간 중 이슈로 보고하는 요구사항은 정해진 형식 없이 다양한 사용자에 의해 작성된다. 개발자가 각 이슈를 파악하여 요구사항을 분류하는 작업은 쉽지 않다. 본 연구에서는 깃허브의 ATOM 프로젝트와 VScode프로젝트의 이슈들에 대하여 텍스트 처리 기법 및 RNN기법을 사용해, 요구사항을 소프트웨어의 특정한 기능으로 분류하기 위한 기반으로 레이블 분류 연구를 진행 하였다. 그 결과 ATOM프로젝트에서는 이슈의 재목과 내용을 모델에 함께 학습 시켰을 때 평균적으로 정밀도 0.56과 재현율 0.37을 달성하였고 VScode프로젝트 에서는 제목만을 학습한 모델이 평균 0.67의 정밀도와 0.59의 재현율을 달성하였다. 또한 기존에 높은 성능을 보인 Naïve Bayes, Decision Tree, Random Forest 모델과의 비교 실험도 진행하였다. 각각의 프로젝트에서 더 높은 성능을 보인 모델이 있었지만 일관되게 RNN기법보다 높은 성능을 보인 모델은 없었다. 그보다 더 주목할만한 결과는 제목을 사용하여 모델을 학습 시켰을 때의 성능이 내용을 사용해 학습 시켰을 때보다 더 성능이 좋았다는 것이다. 이는 핵심어로 구성한 제목이 다양한 설명이 있는 본문보다 분류에는 더 좋은 성능을 낼 수 있다는 해석을 이끌어낸다.

향후 연구로서 우리의 최종 목표는 이슈를 요구사항으로 분류하는 것이다. 6.1절에서 6.3절까지에서 분석한 논의에 따라 편집기 도메인에서 더욱 많은 이슈 데이터를 수집하고, 수집한 데이터에 기반하여 이슈를 자동 요약하여 분류하는 연구를 진행할 예정이다. 상세하게는 6.2절을 보면 요구사항을 분류할 때 전체 내용을 사용하기 보다 요약된 내용을 사용하여 분류하는 것이 더 좋은 성능을 낼 수 있다는 가설을 수립할 수 있다. 따라서 우리는 더욱 우수한 성능의 요구사항 분류 기법 개발을 위해 이슈를 요약하여 요약한 내용을 기반으로 딥러닝 모델을 적용한 분류 연구를 진행할 것이다. 또한 신뢰성을 높이기 위해 좀 더 다양한 프로젝트를 대상으로 많은 양의 데이터를 수집하여 실험을 진행할 것이다. 이를 위하여 6.1절과 6.2절의 가설을 참조하여 이슈를 분류하는 다양한 실험을 진행할 것이다. 최종적으로는 프로젝트에서 요구사항을 도출하고 이슈를 특정 요구사항으로 자동 분류하는 연구와 요약 기법을 통하여 새로운 요구사항을 생성하는 연구로 발전시킬 예정이다.

Reference

[1] Lu, Mengmeng, and Peng Liang. "Automatic Classification of Non-Functional Requirements from Augmented App User Reviews." *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2017. p. 344-353.

[2] Fan, Q., Yu, Y., Yin, G., Wang, T., & Wang, H. "Where is the road for issue reports classification based on text mining?" In *Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on*. IEEE, 2017. p. 121-130.

[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, Vol.9, No.8 pp.1735-1780.

[4] J. Y. Chung, C. Gulcehre, K. H. Cho and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling”, arXiv preprint arXiv:1412.3555, 2014.

[5] Bissyandé, T. F., Lo, D., Jiang, L., Réveillere, L., Klein, J., & Le Traon, Y. "Got issues? who cares about it? a large scale investigation of issue trackers from github." *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*. IEEE, 2013. p. 188-197

[6] Gu, X., Zhang, H., Zhang, D., & Kim, S. "DeepAM: Migrate APIs with multi-modal sequence to sequence learning." *arXiv preprint arXiv:1704.07734* (2017).

[7] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980, 2014.

[8] Guzman, E., El-Haliby, M., & Bruegge, B. "Ensemble methods for app review classification: An approach for software evolution (N)." *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. IEEE, 2015. p. 771-776.

[9] Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. "How can i improve my app? classifying user reviews for software maintenance and evolution." *Software maintenance and evolution (ICSME), 2015 IEEE international conference on*. IEEE, 2015. p. 281-290.

[10] Jindal, Rajni, Ruchika Malhotra, and Abha Jain. "Automated classification of security requirements." *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*. IEEE, 2016. p. 2027-2033.

서비스 기반 V2X 소프트웨어 환경에서의 서비스 소비 품질 최적화 방법

문형철 ○, 백경덕, 고인영
한국과학기술원
hc.moon@kaist.ac.kr, kyeongdeok.baek@kaist.ac.kr, iko@kaist.ac.kr

Optimizing Perceived Quality of Services in Service-oriented V2X Software Environments

Hyeong-Cheol Moon○, Kyeong-Deok Baek, In-Young Ko
Korea Advanced Institute of Science and Technology

요 약

최근 컴퓨팅 기술의 발전에 따라 차량-사물 통신 환경과 같은 대규모 가상 물리 시스템 어플리케이션에 관한 연구가 활발히 이뤄지고 있다. 그 중 차량-사물 통신 환경에 서비스 컴퓨팅을 적용해 차량-사물 통신 환경을 발전하려는 연구가 진행되고 있다. 서비스 기반 차량-사물 통신 환경에서 사용자는 사물 인터넷 기기를 통해 서비스를 전달받는데, 기존의 차량-사물 통신 환경에서는 사용자의 서비스 소비를 네트워크적인 관점에서 바라보고 있다. 그러나 이는 사용자 관점에서 소비되는 서비스의 품질을 보장하지 못한다. 본 논문에서는 서비스 기반의 차량-사물 통신 환경을 정의하고 사용자 관점의 서비스 소비 품질을 위해 서비스 효과도와 서비스 핸드오버를 살펴본다. 또한, 이를 활용하여 사용자의 서비스 소비 계획을 최적화 하는 방법을 제안하고, 이를 기존의 네트워크 관점만을 고려한 방법들과 비교하여 평가한다.

1. 서 론

최근 컴퓨팅 기술의 발전에 따라 소프트웨어의 개념이 기존의 컴퓨터 내에서만 실행되는 프로그램에서 현실 공간까지 아우르는 *가상물리시스템 (Cyber Physical System, CPS)* 수준으로 확장되고 있다. 대표적인 예로 근래의 자동 주행 자동차의 개발과 더불어 *사물 인터넷 (Internet of Things, IoT)*을 활용한 지능적인 *차량-사물 통신 (Vehicle-to-Everything, V2X)* 환경을 구축하려는 움직임이 늘어나고 있다. 일반적으로 CPS 어플리케이션은 사용 가능한 IoT 기기들을 활용하여 스스로 문제를 파악하여 필요한 결정을 내리고 지능적으로 문제를 해결해 나간다.

그러나 실제 환경의 복잡성, 역동성 문제, 그리고 표준화되지 않은 서비스들이 가지는 이종성 문제로 인하여 V2X 환경의 실현은 어려운 문제로 인식되어 왔다. 최근, 이와 같은 V2X 환경의 문제를 해결하기 위해서 V2X 환경의 기기들이 제공하는 기능들을 사용자 관점에서 서비스화 하여 재사용 및 관리를 용이하게 하는 서비스 컴퓨팅 (Service Computing)을 적용하는 연구들이 진행되어 왔다[1].

서비스 컴퓨팅 기반 V2X 환경에서 사용자는 IoT 기기를 통해 전달되는 서비스를 소비하는데, 이때 사용자에게 전달되는 IoT 기기의 물리적 효과의 품질을 *서비스 효과도 (Service Effectiveness)*라고 한다. 또한 서비스 효과도와 사용자의 품질 요구사항에 따라 IoT 기기가 사용자에게 서비스를 전달할 수 있는 공간적 범위를 *서비스 전달 가능 범위 (Service Boundary)*라 한다[2].

V2X 환경에서는 사용자의 이동성(Mobility)이 매우 높기 때문에 서비스를 제공받던 사용자가 서비스 전달 가능 범위를 벗어나게 될 확률이 높다. 따라서 이와 같은 상황에서는 사용자의 움직임에 맞추어 새로운 서비스로 기존 서비스를 대체함으로써 사용자가 IoT 기기들로부터 서비스를 지속적으로 받을 수 있도록 해야 하며, 이 과정을 *서비스 핸드오버 (Service Handover)*라 한다. 그러나 물리적 공간에서 IoT 기기가 사용자에게 실제로 서비스를 전달하는지 여부는 네트워크 연결 여부와는 다른 문제이므로, 단순히 네트워크 상에서 연결 가능한 IoT 기기를 할당하는 것만으로는 서비스 핸드오버를 효과적으로 이룰 수 없다.

서비스 핸드오버 과정에서 서비스를 지속적으로 제공하기 위해 이전 서비스로부터 필요한 사용자의 서비스 관련 데이터를 옮겨야 하는데 이는 서비스 지연 시간을 초래한다. 또한 이는 사용자의 이동 경로상의 네트워크 상태 변화에 따라 데이터 전송 시의 지연 시간이 차이가 날 수 있다. 그러므로 사용자의 서비스 소비 품질을 증가시키기 위해서는 서비스 핸드오버로 인한 서비스 품질 저하를 최소화해야 한다.

본 논문에서는 서비스 기반 V2X 소프트웨어 환경을 사용자의 서비스 소비 관점에서 정의하기 위해 서비스 효과도와 서비스 핸드오버를 살펴보고, 사용자가 V2X 환경에서 서비스를 제공 받을 때 서비스 효과도 고려하여 사용자 관점에서 서비스 전달 품질의 최적화하기 위해 유전 알고리즘을 활용하여 서비스 핸드오버로 인한 서비스 지연 시간을 최소화 하는 방법을 제시한다.

2. 서비스 효과도를 고려한 서비스 소비 계획

V2X 환경에서 사용자는 이동 수단을 타고 도로를 따라 이동한다. 따라서 특정 시점의 V2X 환경을 사용자의 경로와 함께 주변의 IoT 기기들의 집합으로 표현할 수 있다. 사용자는 IoT 기기를 활용하는 서비스를 사용하며 도로를 따라 이동하게 된다. 이때 도로 근처에 수많은 IoT 기기들 중 어떤 IoT 기기를 사용자에게 할당하는지는 서비스 전달 품질에 큰 영향을 끼친다. 사용자에게 IoT 기기를 할당할 때마다 서비스 핸드오버를 실행해야 하는데, 이는 사용자의 서비스 소비 품질을 저해할 수 있다.

사용자에게 높은 품질의 서비스를 제공하기 위해서는 서비스 핸드오버로 인한 서비스 소비 지연 시간을 최소화해야 할 필요가 있다. 실제 환경에서 서비스 핸드오버로 인한 전체 비용을 계산하는 것은 불가능하므로 서비스 핸드오버로 인한 지연 시간을 어떤 사건이 발생할 때까지 걸리는 시간을 표현하는 지수 분포를 활용하여 정의할 수 있다.

V2X 환경에서 사용자의 서비스 소비 지연 시간을 최소화하기 위해 가능한 적은 수의 IoT 기기들을 선택하는 것이 바람직하다. 또한 동시에, 사용할 IoT 기기의 서비스 효과도 값을 서비스 소비 품질에 영향을 주지 않도록 충분한 값 이상으로 유지해야 한다. 이는 사용자 경로와 경로상의 특정 시점의 상태를 바탕으로 사용자가 해당 경로로 움직이며 발생하는 서비스 핸드오버로 인한 지연 시간을 최소화 하는 최적화 문제로 정의된다. 이 최적화 문제는 앞서 언급한 조건을 충족하면서 총 서비스 지연 시간을 최소화 하는 IoT 기기들의 사용 계획을 찾는 문제이다. IoT 기기들의 사용 계획을 배열로 표현한다면 최적화 문제는 배열의 서로 다른 IoT 기기의 사이에 발생하는 서비스 핸드오버로 인한 지연 시간을 최소화 하는 것이 목적이다.

본 논문에서는 유전 알고리즘 (Genetic Algorithm)을 바탕으로 V2X 환경에서 IoT 기기로부터 서비스를 제공받는 사용자가 이동하면서 발생하는 서비스 핸드오버로 인한 서비스 소비 지연 시간을 최소화 하면서, 동시에 서비스 효과도는 적절한 값을 유지하는 서비스 소비 계획 최적화 방법을 제안한다.

3. 실험 및 평가

본 논문에서 제시하는 방법의 비교 평가 실험을 위해 단순히 물리적인 거리만을 고려한 근거리 우선 방식 (Nearest First)과 항상 가장 높은 효과도를 가지는 IoT 기기를 사용하는 효과도 우선 방식 (Max Effectiveness First)를 구현하였다. 실험은 가상 환경에 모든 기기를 무작위로 배치하여 본 논문에서 제안하는 방법과 다른 두 방법의 차이를 시뮬레이션을 통해 비교 실험하였다. 실험은 가상으로 만들어진 무작위 환경에서 주어진 경로를 이동하는 사용자에게 주변의 IoT 기기를 위에서 언급한 3가지 방법을 통해 할당한 후, 서비스 핸드오버로 인한 사용자의 서비스 소비 지연 시간을 측정하였다.

그림 1은 유전 알고리즘과 두 개의 비교 방법을 통해 구한 각 시뮬레이션에서 사용자 소비 계획에서 총 서비스 지연 시간의 비율을 IoT 기기의 밀도 별로 나타낸 그래프이다. 각 알고리즘 간의 서비스 지연 시간을 직접적으로 비교하기 위해 유전 알고리즘과 다른 두 알고리즘에 대한 비율을 비교하여 그 분포를 나타내었다. 그림 1에서 근거리 우선 방식을 통해 구한 서비스 소비 계획의 서비스 지연 시간이 유전 알고리즘보다 모든 IoT 기기의 밀도에 대해 2.5배 정도 길다. 또한 효과도 우선 방식으로 구한 서비스 소비 계획의 서비스 지연 시간도 유전 알고리즘에 비해 2배 이상 길다. 이는 본 논문에서 제안하는 유전 알고리즘이 기존의 알고리즘보다 효과적으로 서비스 핸드오버로 인한 서비스 지연 시간을 줄일 수 있다는 것을 의미한다.

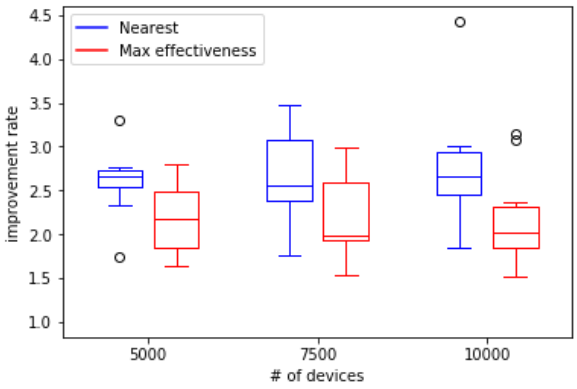


그림 1 유전 알고리즘과 두 비교 방법에 대한 IoT 기기 밀도 별 서비스 지연 시간 비율의 분포 그래프

4. 결론

본 논문에서는 V2X 환경에서 사용자의 서비스 소비 품질을 증가시키기 위해 서비스 기반 V2X 환경과 서비스 효과도를 정의하고 서비스 핸드오버가 V2X 환경에 미치는 영향에 대해 살펴보았다. 그 후에 이를 활용하여 서비스 핸드오버를 고려한 사용자 서비스 소비 품질 최적화 문제를 정의하였다. 해당 최적화 문제의 큰 공간 복잡도를 해결하기 위해 유전 알고리즘을 활용하여 큰 공간 복잡도에서 효율적으로 최적에 가까운 답을 찾아내는 알고리즘을 제안하였다. 그 결과 본 논문에서 제안한 알고리즘을 적용했을 때, 서비스 핸드오버로 인한 서비스 지연 시간이 2배 이상 감소했다.

Acknowledgement

이 성과는 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2016R1A2B4007585).

참고 문헌

[1] Hoang, Dat Dac, Hye-Young Paik, and Chae-Kyu Kim. "Service-oriented middleware architectures for cyber-physical systems." International Journal of Computer Science and Network Security 12.1 (2012): 79-87.

[2] KyeongDeok Baek, and In-Young Ko, “VR-based Simulation for Evaluating Service Quality of Visual Services in IoT Environments”, In Proceeding of 2018 Korea Software Congress (KSC 2018), December 19 - 21, 2018.

캡슐내시경 영상 데이터 중심 소프트웨어 개발을 위한 데이터 셋 품질 평가 매트릭

박예슬[1], 이병정[2], 이정원[1]
[1]아주대학교 전자공학과, [2]서울시립대학교 컴퓨터과학과
yeseuly777@gmail.com, bjlee@uos.ac.kr, jungwony@ajou.ac.kr

Quality Evaluation Metric of Data Set to Develop Data-driven Software for Capsule Endoscopic Images

Ye-Seul Park[1], Byungjeong Lee[2], and Jung-Won Lee[1]
[1]Department of Electrical and Computer Engineering, Ajou University.
[2]Department of Computer Science and Engineering, The University of Seoul.

요 약

최근 하드웨어의 처리 속도가 향상됨에 따라 빅데이터를 응용한 소프트웨어에 대한 관심이 늘어나고 있다. 이에 대한 원인은 데이터를 학습하여 얻은 패턴이나 관계가 보다 많은 증거로 인해, 신뢰성 있는 성능을 보장받을 수 있기 때문이다. 하지만 일각에서는 무분별하게 축적되고 있는 데이터에 대해 회의적인 의견을 보이는데, 이는 축적된 데이터의 품질을 보장할 수 없기 때문이다. 데이터 품질 문제는 영역 지식에 대한 이해가 어려운 의료 분야의 경우 더욱 크다. 의료 영상 데이터의 경우, 영상에서 확인되는 해부학적 요소들에 대한 이해를 통해 레이블링 작업을 수행해야 한다. 그러나 일반적인 소프트웨어 개발자가 이를 직접 수행하기에는 어려움이 있어 수집한 데이터 중 대부분이 학습에 사용할 수 없게 되기 때문이다. 따라서 본 논문에서는 의료 영상 데이터 중 하나인 캡슐내시경 영상의 품질을 평가할 수 있는 매트릭을 제안하고자 한다. 제안하는 매트릭은 크게 세 가지 지표(데이터 정상성, 학습 적합성, 해부학적 완전성)로 이루어져 있으며, 이를 기반으로 11명의 환자 데이터를 세 가지 품질 등급(고품질, 중품질, 저품질)로 구분하여 학습을 진행하였다. 학습을 수행한 결과, 고품질의 데이터에서는 0.96, 중품질 0.94, 저품질 0.87의 정확도를 보였으며, 이는 데이터 셋의 품질이 소프트웨어의 성능에 미칠 수 있는 영향을 분석할 수 있게 하였다.

1. 서론[1]

IT 산업의 발달로 인하여, 다양한 형태의 데이터가 생성되고 있다. 특히, 최근에는 데이터가 방대한 양으로 축적되어 빅데이터를 이루고 있는데, 하드웨어의 처리 속도가 향상됨에 따라 이와 같은 빅데이터를 응용한 소프트웨어(Data-driven Software)들이 개발되고 있는 상황이다. 빅데이터가 주목을 받고 있는 이유는 데이터를 학습하여 얻은 패턴이나 관계가 보다 많은 증거(Evidence)에 의해 신뢰성 있는 성능을 보장받을 수 있기 때문이다. 하지만 일각에서는 무분별하게 축적되고 있는 데이터를 무조건적으로 사용하는 것에 회의적인 의견이 있다. 그 이유는 무분별하게 축적된 빅데이터의 경우, 데이터의 품질을 보장할 수 없기 때문이다.

데이터 품질(Data Quality)이란 “데이터의 최신성, 정확성, 상호 연계성 등을 확보하여 사용자에게 유용한 가치를 줄 수 있는 수준”이라는 뜻으로[1], 수집된 데이터를 활용하여 창출될 수 있는 가치를 나타내는 지표라고 할 수 있다. 즉, 데이터를 학습 하는 경우에는 학습을 수행하는 개발자에게 해당 데이터를 활용하여 도출될 수 있는 유의미한 패턴이나 관계의 수준이 데이터의 품질을 결정할 수 있다. 예컨대, 영상 데이터를 학습하여 특정 항목에 대한 분류기를 개발하였다면, 학습 데이터 셋이 분류기(Classifier)의 성능에 미치는 영향력이 데이터 품질로 측정될 수 있다. 이 때, 레이블(Label)이 부여된 데이터 만이 학습에 활용될 수 있기 때문에, 레이블이 부여된 데이터가 전체 데이터 셋의 품질에 큰 영향을 미칠 수 있게 된다.

레이블이란 데이터가 가지고 있는 의미에 대한 카테고리 형태의 정의를 말하며, 개개의 데이터에 레이블을 부여하는 작업을 레이블링(Labeling)이라 한다. 이러한 레이블링 작업은 의료 분야에서 큰 문제가 될 수 있다. 일반적인 소프트웨어 개발자가 의료 데이터를 이해하는데 많은 영역 지식(Domain Knowledge)과 의료적인 노하우(기술)가 요구되어, 레이블링 작업을 위한 오버헤드(Overhead)가 발생하기 때문이다.

1) 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00309)
2) 이 논문은 2018년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. NRF-2014M3C4A7030504).

[2]에서는 망막 안저 사진(Retinal Fundus Photographs)을 이용하여 당뇨병 망막증(Diabetic Retinopathy)을 검출하는 딥러닝(Deep Learning)기법이 제안되었다. 특히, [2]에서는 약 54명의 안과 의사 혹은 안과 레지던트(US Licensed Ophthalmologists and Ophthalmology Senior Residents)로부터 약 1년간 검증받은 고품질의 데이터 셋을 사용하였다.

데이터 품질을 보장하는 것은 고품질의 소프트웨어를 개발할 수 있는 기반이 된다. 그러나 [2]와 같은 방식으로 고품질의 데이터 셋을 확보하는 것은 많은 시간과 비용이 들게 된다. 또한, 현재 많은 개발자들은 고품질의 데이터 수집은 고사하고 수집된 데이터 셋의 품질 조차도 가늠하지 못한 채 학습을 수행하고 있다. 달리 말하면 많은 개발자들은 본인이 가지고 있는 데이터 셋이 학습이 불가능한 수준인지 모른 채 학습을 진행하고 있다는 것이다. 따라서 본 논문에서는 의료 영상 데이터를 이용하여 학습을 진행할 때, 수집된 데이터 셋에 대한 품질을 평가할 수 있는 매트릭을 제안하고자 한다. 이와 같은 데이터 셋의 품질 평가는 수집된 데이터 셋의 품질을 보장할 수 있는 기반이 될 수 있으며, 더 나아가 고품질의 소프트웨어를 개발할 수 있는 기초 작업이 될 수 있다.

수집 데이터 셋에 대한 품질 평가 매트릭을 설계하기 위해서는 세부적으로 평가를 진행할 수 있는 적절한 도메인의 선정이 필요하다. 의료 분야의 경우, 도메인에 따라 주목해야 하는 영상 데이터의 특성이 상이하기 때문이다. 이를 고려하여, 본 연구에서 주목한 영상 모달리티(Modality)는 캡슐내시경이다. 캡슐내시경의 경우, 다른 의료 영상 데이터와는 다르게 입에서 항문까지의 위장관 구조를 환자마다 균일하게 촬영한다. 예를 들면, 입, 식도, 위, 십이지장, 공장, 대장 등의 위장관 구조가 환자마다 비슷한 양상으로 촬영될 수 있다. 대조적으로, 혈관조영영상(Angiography)의 경우 촬영 되는 각도에 따라 관찰될 수 있는 혈관 구조가 다르다. 이와 같은 캡슐내시경의 특성은 수집된 데이터 셋에 대해 균일한 평가 기준을 적용할 수 있게 하므로, 본 연구에서는 캡슐내시경 영상 데이터를 기준으로 품질 평가 매트릭을 설계하였다.

제안하는 매트릭은 세 가지 기준(데이터 정상성, 학습 적합성, 해부학적 완전성)으로 구성되어 있으며, 그림 1은 이를 기반으로 평가된 결과를 도식화 한 것이다. 첫째, 데이터 정상성(Data Nomality)은 수집된 데이터 셋 중에서 학습에 활용될 수 있는 데이터의 비율을 말한다. 이는 실제로 학습에 활용될 수 있는 유효한 데이터의 비율을 의미하게 된다. 둘째, 학습 적합성(Learning Fitness)은 전체 데이터 셋 중에서 레이블이 되어 있거나 레이블링을 할 수 있는 데이터 셋의 비율을 의미한다. 만약, 레이블링을 수행할 수 없다면 학습에 활용할 수 없기 때문이다. 마지막으로, 해부학적 완전성(Anatomical Completeness)이란 캡슐내시경에서 관찰되어야 하는 주요 해부학적 요소들 중에서 실제로 관찰된 요소의 비율을 의미하는 것으로, 필수적으로 관찰되어야 하는 요소의 비율과 선택적으로 관찰되어야 하는 요소의 비율이 합해져 측정될 수 있다. 예컨대, 병변(예: 용종, 출혈 등)은 위장관에서 관찰될 수 있지만 건강한 사람의 경우 필수적으로 관찰될 필요가 없기 때문에 선택적인 요소에 해당된다. 그림 1의 예제로 살펴본다면, 데이터 정상성의 경우 약 92%, 학습 적합성 92.3%, 해부학적 완전성 95%(Selective Factor 0.95 기준)으로 평가됨을 확인할 수 있다.

본 논문에서는 설계된 품질 평가 매트릭을 기반으로 수집된 11명의 캡슐내시경 영상 데이터 셋에 대해 평가하였다. 또한 환자 별 품질 평가 결과를 기반으로 세 가지 품질 등급(고품질, 중품질, 저품질)에 해당되는 데이터 셋을 생성하였으며, 각각의 데이터 셋을 사용하여 학습을 진행하였다. 학습목표는 “위장관 교차점 (Gastrointestinal Tract Junction)” 이라고 하는 해부학적 요소(필수적)를 검출하기 위해 진행되었다. 학습을 수행한 결과, 고품질의 데이터에서는 0.96, 중품질 0.94, 저품질 0.87의 정확도를 보였다. 이와 같은 결과는 고품질의 데이터 셋이 학습 성능에 미치는 영향이 크다는 것을 확인할 수 있게 한 것으로, 데이터 셋의 품질에 대한 중요성을 나타내었다. 향후에는 제안하는 품질 평가 매트릭을 기반으로 보다 나은 양질의 데이터 셋을 보강할 수 있게 하고, 위장관 교차점 분류기의 성능을 고도화할 예정이다.

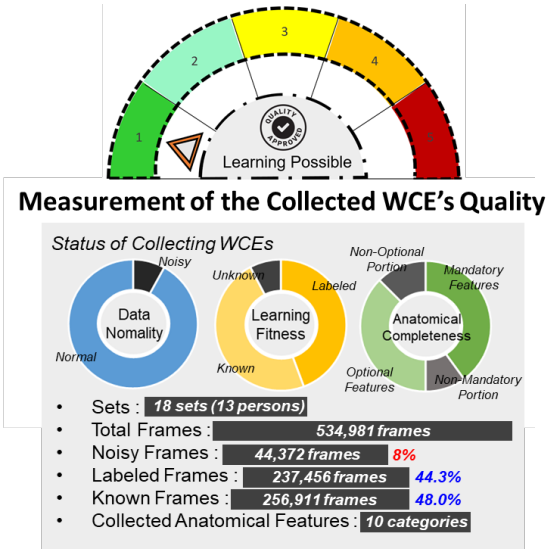


그림 1. 캡슐내시경 영상 데이터 셋 품질 평가 예상도

2. 관련 연구

2장에서는 본 연구에 대한 관련 연구를 소개하고자 한다. 2.1절에서는 의료 영상 데이터 셋의 품질 평가 방법, 이어 2.2절에서는 캡슐내시경 영상의 학습과 관련된 연구 동향에 대해 구분하여 소개된다.

2.1 의료 영상 데이터 셋의 품질 평가

2.1절에서는 의료 영상 데이터 셋의 품질 평가 방법에 대해 소개하고자 한다. 데이터 품질을 평가하기 위한 방법은 대상 데이터의 구조화된 정도에 따라 다를 수 있다. 예를 들면, 전자의무기록(EMR, Electronic Medical Record)과 같은 정형 데이터 형태에서는 데이터의 정의(Definition), 수집(Collection), 처리(Processing), 표현(Representation), 사용(Use)과 같은 항목이 의료 데이터 품질의 영향 요소가 된다. 또한, 정확도(Accuracy), 현재성(Currency), 완전성 (Completeness), 일치성(Consistency), 적시성(Timeliness), 관련성(Relevance), 입자성(Granularity), 특이성 (Specificity), 정밀성(Precision), 속성(Attribute)이 품질 평가 항목으로써 활용된다[1]. 그러나 구조화되어 있지 않은 의료 영상 데이터(비정형 데이터)의 경우, 데이터의 형태가 불규칙하게 이루어져 있어 이와 같은 기준을 적용할 수 없다. 따라서 영상 데이터는 분야에 따라 기준을 다르게 하여 품질을 평가하고 있다. [3]에서는 베이지안(Bayesian) 기법을 적용하여 CT (Computed Tomography) 영상의 품질을 평가하는 방식을 제안하였으며, [4]에서는 미숙아의 망막 병증을 진단할 때 활용될 수 있는 저부 이미지(Fundus Images)의 품질을 평가하기 위한 딥러닝 기반의 방법론을 제시하였다. 또한, [5]에서는 영상의 품질 평가 결과를 기반으로 초음파 영상의 유사성을 판단하는 연구를 진행한 바 있다. 그러나 대부분 단일 영상의 품질을 평가하는 데 초점을 맞추고 있다. 따라서 현재로서는 수집된 영상 데이터를 학습에 적용할 때, 발현될 수 있는 데이터 셋의 품질을 직접적으로 평가하기 어렵다.

2.2 캡슐내시경 영상 데이터를 이용한 학습

2.2절에서는 캡슐내시경 영상 데이터를 활용하여 학습을 수행한 연구들에 대해 소개한다. 일반적으로 캡슐내시경 영상은 위장관 내의 병변을 관찰하기 위해 사용된다. 이 때, 발견될 수 있는 병변의 종류는 무수히 많지만, 현재 학습이 가능한 것은 병변의 형태가 특이하거나 색감이 구별되기 쉬운 경우이다. 예를 들어, [6]에서는 폴립(Polyp)이라는 융기성 병변을 검출하기 위해 딥러닝 기법을 적용하였다. [6]에서 학습 목적으로 삼은 폴립의 경우, 위장관 점막에 튀어 나와 있기 때문에 정상일 때에 비해 형태가 구별되기 쉬운 사례이다. 병변의 색감이 중요한 검출 지표가 되는 경우도 있다. [7]에서는 위장관 출혈(Hemorrahge) 이미지를 검출 하기 위해, [8]에서는 혈흔(Bleeding) 이미지를 검출 하기 위해 CNN(Convolutional Neural Network)을 적용하였다. [9]에서는 [8]과 마찬가지로 혈흔 이미지를 검출하기 위한 연구를 제안하였는데, 전통적인 기법을 도입하여 획득한 특징과 CNN을 적용하여 획득한 자동 생성 특징을 결합하여 학습에 적용하였다는 점에서 차이점을 보였다. 이 외에도 [10]에서는 위장관 내강에 존재하는 물질(Contents)를 검출하여 위치(식도, 위, 소장, 대장)를 추적하기 위한 학습을 진행하였다. 이와 같은 연구에서 공통적으로 언급하는 것은 학습에 사용한 데이터 셋에 대한 평가이다. 각 연구에서는 학습에 사용된 데이터의 품질을 자체적으로 평가하여 사용하였으며, 이는 학습을 진행하는 많은 연구에서 데이터 셋의 품질 평가를 필요로 한다는 것을 보여주고 있다.

3. 캡슐내시경 데이터 셋 품질 평가 매트릭

3장에서는 캡슐내시경 데이터 셋의 품질을 평가하기 위한 매트릭을 제안한다. 본 논문에서는 그림 2와 같이, [1]에서 제시하고 있는 5가지의 품질 영향 요소와 10가지의 품질 평가 항목을 기반으로, 캡슐내시경 데이터 셋의 품질 평가 매트릭을 설계하였다. 먼저, 품질 평가 요소에 대해 살펴보자. 본 연구에서는 학습을 위한(Data Use) 캡슐내시경 수집(Data Collection)을 목적으로 하고 있다. 또한, 이와 같은 데이터를 학습에 활용하기 위해, 데이터 레이블링 부분에 초점을 맞추어 매트릭이 설계된 상황이다(Data Definition). 점선으로 표시된 데이터 처리(Data Processing), 데이터 표현(Data Representation)의 경우, 학습을 수행하기 위해 항상 영상 처리 과정을 거쳐야 하는 것은 아니기 때문에 설계에 반영되지 않았다. 다음은 품질 평가 항목이다. 정상적인 데이터의 수집(데이터 정상성)에 대한 부분은 정확성, 완전성을, 레이블링 데이터에 대한 비율(학습 적합성)은 관련성을 기반으로 설계되었다. 또한, 데이터에 내재된 의미를 고려하는 부분(해부학적 완전성)은 입자성, 특이성, 정밀성, 속성을 기반으로 하였다. 점선으로 표시되어 있는 현재성, 일치성, 적시성의 경우, 데이터의 갱신에 관련된 부분인데, 본 연구에서는 영상 데이터를 가공하는 부분은 고려하고 있지 않으므로, 설계에 사용되지 않았다. 본 연구에서는 구체적인 매트릭은 표 1과 같으며, 매트릭을 구성하고 있는 세 가지 기준(정상성, 학습 적합성, 해부학적 완전성)에 대한 설명은 다음과 같다.

3.1 데이터 정상성(DN: Data Nomality)

첫 번째 기준은 데이터 정상성이다. 데이터 정상성이란 수집된 데이터 셋 중에서 학습에 활용될 수 있는 정상적인 데이터의 비율을 의미한다. 여기서 정상적인 데이터와 대조되어 사용될 수 있는 단어는 학습 노이즈다. 학습 노이즈란 데이터를 학습할 때, 학습에 방해가 될 수 있는 데이터를 말하는 것으로 캡슐내시경의 경우, 좋은 해상도를 지닌 데이터일지라도 학습 노이즈가 될 수 있다. 예를 들면, 지나치게 거품이 많이 끼어 있어 위장관 내부를 살펴보기 어렵거나, 장액이나 이물질이 많아 영상을 판별하기 어려운 경우가 있다. 본 논문에서는 이와 같은 학습 노이즈

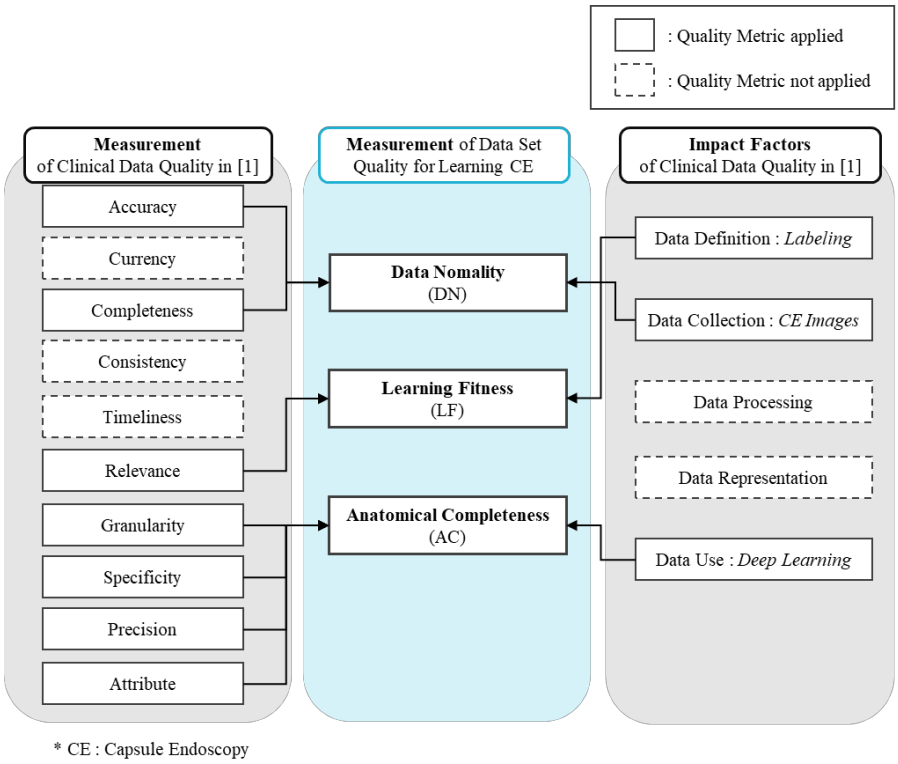


그림 2. [1]의 품질 평가 요소 기반 캡슐내시경 데이터 셋 품질 평가 매트릭 설계

표 1. 캡슐내시경 데이터 셋 품질 평가 매트릭 정의

매트릭 이름 (Metric Name)	매트릭 정의 (Metric Definition)	개념 수식 (Conceptual Formula)
데이터 정상성 DN (Data Nomality)	수집된 데이터 셋 중에서 정상적인 데이터가 얼마나 수집되었는가?	$\mathrm{DN} = \frac{Norm}{T} \quad (Norm = 1 - \sum_i N_i)$ * *Norm* = 학습에 도움이 되는 데이터(프레임) 수 * N_i = 학습에 방해가 되는 데이터(프레임) 수 이 때, i는 학습의 방해가 되는 데이터의 종류를 의미
학습 적합성 LF (Learning Fitness)	수집된 데이터 셋 중에서 학습을 위한 레이블링 작업이 가능한 데이터가 얼마나 수집되었는가?	$\mathrm{LF} = \frac{L}{T}$ $(L = T - NL(or\ \ LUI) = LLP + LLN + LUP)$ * *NL* = 학습 불가능 프레임(Not Learnable Frames) 수 * *LLP* = 양성으로 레이블링 된 프레임(Frames Labeled to Positive) 수 * *LLN* = 음성으로 레이블링 된 프레임(Frames Labeled to Negative) 수 * *LUP* = 레이블링 가능한 프레임(Frames Unlabeled but Possible) * *LUI* = 레이블링 불가능한 프레임 (Frames Unlabeled and Impossible) 수

표 2. 캡슐내시경 데이터 셋 품질 평가 매트릭 정의 *(Continued)*

매트릭 이름 (Metric Name)	매트릭 정의 (Metric Definition)	개념 수식 (Conceptual Formula)
해부학적 완전성 **AC** (Anatomical Completeness)	수집된 데이터 셋을 해부학적으로 분석하였을 때, 획득한 해부학적 요소들이 얼마나 완전하게 수집되었는가?	$\mathrm{AC} = \alpha \times \frac{MF}{T_M} + (1-\alpha) \times \frac{OF}{T_O}$ * MF = 관찰된 필수적인 해부학적 요소(Mandatory Anatomic Features) 수 * OF = 관찰된 선택적인 해부학적 요소(Optional Anatomic Features) 수 * α = 학습 목적 선택적 요인(Selective Factor)
* T = 전체 데이터 수, T_M = 필수적인 해부학적 요소의 전체 수(10), T_O = 선택적인 해부학적 요소 전체 수(29)		

영상을 배제하고 실제 학습에 활용될 수 있는 데이터를 정상 데이터라 하였으며, 전체 데이터 셋 중에서 정상 데이터가 차지하는 비율을 데이터 정상성이라고 정의하였다. 이 때, 배제될 수 있는 학습 노이즈 영상의 분류는 그림 3과 같다. 실선으로 표시된 경우는 항상 발견 가능한 학습 노이즈 이고, 점선으로 표시된 경우는 상대적으로 발견 가능한 학습 노이즈를 나타낸다.

3.2 학습 적합성(LF, Learning Fitness)

두 번째 기준은 학습 적합성이다. 의료 영상 데이터의 경우, 다른 분야와 달리 개발자가 직접 학습 데이터 레이블링을 수행하기 어렵다. 레이블링을 하기 위해 촬영된 부분에 대한 해부학적 의미를 분석해야 하는데, 이는 많은 의료 도메인 지식을 요구하기 때문이다. 따라서 의료 영상을 학습하는 대부분의 연구에서는 레이블링이 되어 있는 데이터에 대한 수요가 높다. 학습 적합성은 이와 같은 레이블링 데이터에 대한 비율을 의미하며, 이를 구하기 위해 분석되어야 하는 항목은 표 3과 같다.

먼저, 학습 데이터 셋은 크게 레이블링이 되어 있는 경우와 되어 있지 않은 경우로 구분할 수 있다(대분류). 보통, 레이블링이 되어 있는 몇 개의 프레임을 제외하고, 대다수의 프레임은 레이블링이 되어 있지 않아 추가적인 레이블링 작업이 필요하다. 다음은 레이블링 된 경우, 되지 않은 경우에 대한 세부 분류이다(소분류). 레이블링이 되어 있는 경우, 학습 목적에 부합하는 양성 레이블링이 된 경우와 학습 목적에는 부합하지 않지만 비교군으로 활용될 수 있는 음성 레이블링이 된 경우로 구분된다. 또한, 레이블링이 되어 있지 않은 경우는 레이블링 된 데이터를 기반으로 개발자가 레이블링을 할 수 있는 경우와 전혀 레이블링을 할 수 없는 경우로 나뉜다. 예컨대, 캡슐 내시경 영상의 경우 식도와 위의 교차점인 위식도 접합부(Z-Line), 위와 소장의 교차점인 유문판(Pyloric Valve)이 레이블링이 되어 있다면, 그 사이에 존재하는 데이터는 위(Stomach)로 레이블링 할 수 있다. 이와 같은 경우는 레이블링이 되어 있지 않지만, 개발자가 레이블링을 할 수 있는 경우(LUP)에 속하게 되며, LUI는 이조차도 어려운 경우를 의미한다. 데이터 분류군 중에서 학습에 활용될 수 있는 경우는 양성 레이블링이 된 경우(LLP), 음성 레이블링이 된 경우(LLN), 레이블링이 가능한 경우(LU-P)가 해당된다. 즉, 학습 적합성이란 전체 데이터 셋 중에서 학습에 활용될 수 있는 데이터(LLN, LLN, LUP)의 비율을 의미한다.

3.3 해부학적 완전성(AC, Anatomical Completeness)

마지막으로 세 번째 기준은 해부학적 완전성이다. 캡슐내시경의 경우 인체의 위장관을 촬영하는 기기이기 때문에, 촬영된 영상에는 위장관에서 관찰되어야 하는 필수 해부학적 요소들이 존재해야 한다. 해부학적 완전성이란 이와 같은 부분에서 착안한 품질 평가 지표다. 품질이 좋은 데이터에서는 관찰되어야 하는 주요 해부학적 요소들이 대부분 관찰 가능하다. 하지만 품질이 나쁜 데이터에서는 학습 노이즈에 의한 영향이나 다른 환경적인 요인으로 인해 관찰 불가능한 경우가 있다.

앞서 예시로 들었던 유문판(Pyloric Valve)이 이에 대한 하나의 예제가 될 수 있다. A라는 환자로부터 획득한 데이터 셋에서는 위에서 소장으로 넘어가는 유문판이 선명하게 촬영되었다. B라는 환자의 영상에서는 캡슐이 위에 정체되어 위에서 관찰되는 유문판이 촬영되었지만 그 이후(예: 소장)의 영상 데이터는 관찰할 수 없다. 마지막으로 C라는 환자의 영상에서는 유문판이 촬영되었지만, 거품과 이물질이 많이 끼어 있어 정확하게 판별하기 어렵다. 만약, A-C 환자의 데이터 셋을 이용하여 유문판을 학습하고자

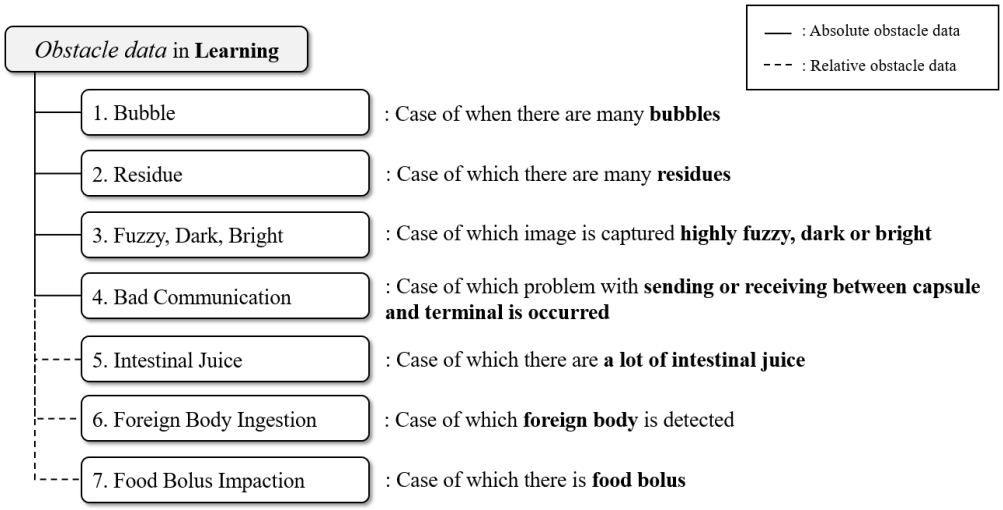


그림 3. 캡슐내시경 영상 데이터 학습 노이즈 분류

표 3. 학습 데이터 셋의 레이블링 관점 분석 기준

구분 (L/NL)	분류(Classification)	
	대분류 (Category)	소분류 (Sub-category)
L	레이블링이 된 경우(Labeled)	양성 레이블링 된 경우 (Labeled to Positive, LLP)
		음성 레이블링 된 경우 (Labeled to Negative, LLN)
	레이블링이 되지 않은 경우(Unlabeled)	레이블링 가능한 경우 (Possible to label, LUP)
NL		레이블링 불가능한 경우 (Impossible to label, LUI)
* L: Learnable (학습 가능)		
* NL: Not-Learnable (학습 불가능)		

하는 개발자라면, A는 학습 목적에 부합하는 양성 데이터(유문판)와 음성 데이터를 모두 갖는 고품질의 데이터일 것이다. 하지만 B의 경우는 양성 데이터, 음성 데이터 모두 일부만 획득한 중품질의 데이터가, C의 경우는 양성 데이터 중 학습에 사용할 수 있는 데이터가 매우 적은 저품질의 데이터가 될 것이다.

그러나 위장관에서 관찰될 수 있는 모든 해부학적 요소들이 항상 관찰 가능한 것은 아니다. 예를 들면, 위장관 점막에서 관찰 가능한 융기성 병변(예: 폴립, 용종 등)의 경우 건강한 사람에게서는 관찰되지 않는다. 따라서 병변과 같은 해부학적 요소는 필수적인 요소가 될 수 없다. 이를 반영하여, 본 논문에서는 위장관에서 관찰될 수 있는 해부학적 요소를 크게 필수적인 요소(MF), 선택적인 요소(OF)로 구분하였으며 해부학적 완전성은 전체 관찰 가능한 해부학적 요소의 개수 중에서 관찰된 요소의 비율로 판단할 수 있다. 이 때, 학습의 목적에 따라 필수 요소와 선택 요소의 중요성이 달라질 수 있으며, 본 논문에서는 선택적 요인(Selective Factor)을 활용하여 이를 조절할 수 있게 하였다. 해부학적 요소에 대한 세부적인 설명은 그림 4와 같으며, 이는 캡슐내시경 관련 표준인 [11-12]를 기반으로 정의되었다.

4. 캡슐내시경 데이터 셋 품질 별 학습 성능 분석

4장에서는 3장에서 정의된 품질 매트릭을 기반으로 실제 캡슐내시경 데이터 셋의 품질을 분석하고, 위장관 교차점 인식에 관한 학습 네트워크를 설계하여 품질에 따른 학습 성능을 비교해본다.

4.1 캡슐내시경 데이터 전처리: 데이터 생성

의료 영상 데이터는 병원 내의 의료영상전송장치 소프트웨어(PACS: Picture Archiving and Communicating Software)를 사용하여 관리된다. 그러므로 병원으로부터 수집된 캡슐내시경 영상 데이터는 의료 영상 데이터 전용 뷰어(Viewer)를 이용해야만 조회할 수 있다. 그러나 이와 같은 파일 형식은 학습용 데이터에 적합하지 않기 때문에, 다음과 같은 절차에 따라 학습용 데이터를 생성하는 과정이 필요하다.

① 의료 영상 전용 소프트웨어를 이용한 데이터 조회
: 먼저, 의료영상전송장치 소프트웨어(PACS)에 연동되어 있는 의료 영상 전용 뷰어(Viewer)를 이용하여 수집된 데이터를 열고 조회한다. 소프트웨어 종류 및 버전에 따라 약간의 차이는 있겠지만, 의료 전문가가 직접 작성한 주석(annotation), 레포트(report) 등과 같이 레이블링에 활용될 수 있는 정보를 얻을 수 있다. 또한, 12~14 시간 정도의 재생 시간을 보유한다.

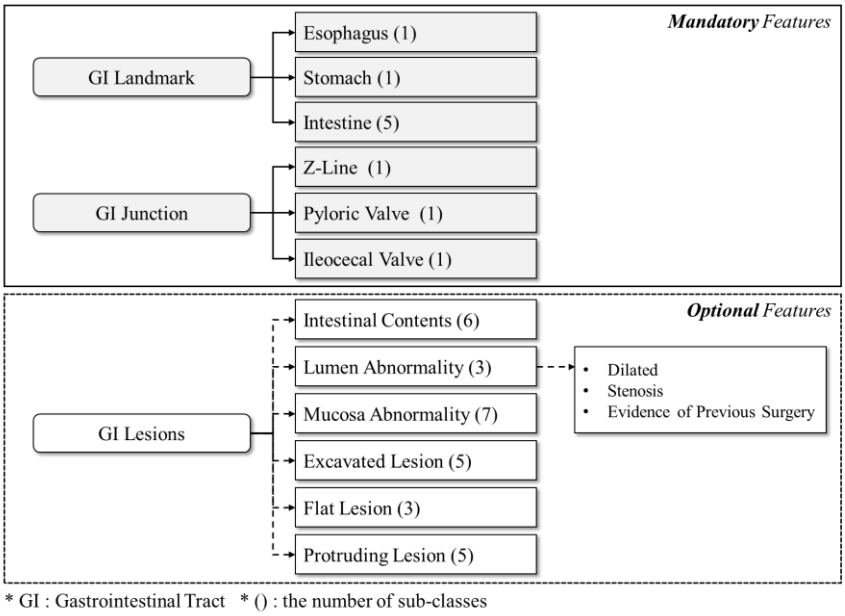


그림 4. 캡슐내시경 영상의 발견 가능한 해부학적 요소

② 영상(비디오) 형태의 데이터 추출
: 다음은 영상(예: .mpg) 형태의 데이터를 추출하는 과정이다. 의료 영상 전용 뷰어(Viewer)에서 영상 추출 기능을 제공하는 경우가 있으나, 그렇지 않은 경우도 있기 때문에 직접 영상을 추출하는 과정이 필요할 수 있다. 이 과정에서 획득한 영상의 경우, 1~2 시간 정도의 재생 시간을 보유하는데, ①의 시간과 차이가 나는 이유는 의료 영상 뷰어와 일반 동영상 뷰어 간의 초당 프레임 별 디코딩 비율이 차이가 나기 때문이다. 실제로, ①과 ②에서의 총 프레임 개수는 동일한 것을 확인할 수 있었다.

③ 프레임(이미지) 형태로 데이터 분할
: 다음은 ②에서 획득한 영상 데이터를 프레임(.png 혹은 .jpg 등)의 이미지 형태로 분할하는 과정이다. 이 과정에서는 다양한 오픈 소스를 활용할 수 있으며, 본 연구에서는 opencv-python 라이브러리를 활용하였다.

④ 중복 프레임(이미지) 제거
: 마지막으로 ③에서 획득한 프레임에서 중복되어 촬영된 프레임을 제거하는 과정을 거친다. 중복을 제거하는 방식은 프레임 내 픽셀 별 값이 모두 동일한 프레임을 중복 프레임으로 추출하는 방식으로 진행되었다. 표 4와 같이 중복 제거 비율은 20%로 고정하여 제거하였다.

총 11명의 환자에 대한 캡슐내시경 영상을 수집하였으며, ①~④의 과정을 통해 생성된 데이터는 표 4와 같이 요약된다. 본 연구에서는 이와 같은 4단계를 거친 데이터(④의 결과)를 최종 학습용 데이터로 활용하였다.

표 4. 환자 별 학습용 데이터 생성 현황

번호	학습용 CE 데이터 생성 절차				
	①	②	③	④	중복 제거 비율
	의료SW 재생시간	일반SW 재생시간	프레임 개수	프레임 개수	
1	13:52:40	1:26:00	128,955	25,791	20%
2	12:26:19	0:33:01	49,490	9,898	20%
3	13:15:37	1:05:27	98,130	19,626	20%
4	15:00:39	0:55:15	82,840	16,568	20%
5	12:10:11	1:05:26	98,110	19,622	20%
6	14:20:57	2:07:00	190,450	38,090	20%
7	15:07:32	1:03:38	95,400	19,080	20%
8	9:20:56	0:59:15	88,840	17,768	20%
9	14:06:32	1:29:23	134,030	26,806	20%
10	13:46:03	1:40:16	150,350	30,070	20%
11	14:38:04	1:38:58	148,420	29,684	20%

4.2 캡슐내시경 데이터 셋 응용: '위장관 교차점 인식'

본 연구에서는 의료 영상 기반 진단 보조 프로그램 개발을 위해, "위장관 교차점 인식"이라는 학습 목적을 선정하였다. 위장관 교차점은 식도, 위, 소장, 대장과 같은 주요 기관(Organ)을 지나갈 때, 교차 지점에서 발견되는 해부학적 요소를 말한다(식도위접합점, 유문판, 회맹판). 이는 건강한 사람이나 환자에게 모두 관찰 가능한 요소로, 필수적인 해부학적 요소가 더 중요하게 된다. 따라서 본 연구에서는 품질 평가 지표를 위한 선택 요인을 0.95로 정의하여 평가하였다. 환자 별 학습용 데이터에 대한 품질 평가 결과는 표 5와 같다.

① 데이터 정상성 평가
: 먼저, 데이터 정상성은 전체 프레임 중에서 정상 프레임이 차지하는 비율을 의미한다. 환자 11명에 대해 분석한 결과 최소 0.64에서 최대 1.00까지의 결과를 얻을 수 있었으며, 평균 0.92로 평가되었다.

② 학습 적합성 평가
: 다음은 학습 적합성에 대한 평가 결과이다. 학습 적합성은 전체 프레임 중에서 양성 혹은 음성으로 레이블링이 되어 있거나 가능한 데이터의 비율을 의미한다. 평가 결과, 최소 0.12에서 최대 0.99, 평균 0.87의 값을 보였다.

③ 해부학적 완전성 평가
: 마지막으로 해부학적 완전성에 대한 평가 결과이다. 평가 결과, 최소 0.38, 최대 0.95, 평균 0.78의 결과를 보였다.

특히, 환자 5의 경우는 데이터 정상성 0.96, 학습 적합성 0.96을 보였지만, 해부학적 완전성은 0.57로 적은 수치를 보였다. 또한, 환자 11의 경우 데이터 정상성은 1.00, 학습 적합성은 0.99를 보였지만 해부학적 완전성의 경우 0.48을 보였다. 이와 같은 양상은 수집된 데이터 전체가 학습 노이즈가 적고, 레이블링이 불가능한 데이터가 적을지라도, 관찰되어야 하는 필수 요소들이 결여되어 수집된 것을 의미하게 된다. 본 연구에서는 이와 같은 품질 평가 결과가 실제 학습에 미치는 영향을 확인하기 위해, 학습 네트워크를 설계하였다. 가장 먼저 진행된 내용은 학습 데이터 셋에 대한 생성이다. 본 연구에서는 품질 별 데이터 셋(고품질, 중품질, 저품질)을 생성하기 위해 그림 5와 같이 진행하였으며, 세부적인 설명은 다음과 같다.

① 총 데이터 수 기반 평가용 데이터의 선별
: 평가용 데이터(test data)란 학습에 사용되지 않고, 학습된 네트워크의 평가만을 위해 사용되는 데이터를 의미한다. 평가용 데이터의 경우, 너무 적은 양으로 평가를 수행할 수 없기 때문에, 학습에 활용되는 전체 프레임의 양이 많은 3명의 환자(6, 10, 11)을 선별하였다.

② 학습용 데이터와 평가용 데이터의 분리
: 일반적으로 학습을 진행할 때, 학습용 데이터(training data), 검증용 데이터(validation data), 평가용 데이터(test data)를 구분한다. 그러나 수집된 환자의 샘플이 적어, 학습용 데이터와 평가용 데이터 두 가지 군으로 분류하여 데이터 셋을 구성하였다. 평가용 데이터는 ①에서 선별된 각각의 환자 데이터로 선정하였고, 학습용 데이터의 경우 이를 제외한 나머지 데이터로 정하였다.

③ 각 데이터 셋에 대한 품질 평가
: 마지막으로 선정된 세 가지 데이터 셋을 품질별로 분류하는 작업을 진행한다. 이 때, 4.1절에서 분석한 품질 분석 결과의 평균 값을 활용하였다. 분석 결과 학습용 데이터 셋과 평가용 데이터 셋의 평균 품질 평가 값을 기반으로 고품질, 중품질, 저품질의 데이터 셋을 생성하였다.

④ 각 데이터 셋에 대한 품질 평가
: 마지막으로 선정된 세 가지 데이터 셋을 품질별로 분류하는 작업을 진행한다. 이 때, 4.1절에서 분석한 품질 분석 결과의 평균 값을 활용하였다. 분석 결과 학습용 데이터 셋과 평가용 데이터 셋의 평균 품질 평가 값을 기반으로 고품질, 중품질, 저품질의 데이터 셋을 생성하였다.

표 5. 환자 별 학습용 데이터 품질 평가 결과

번호	전체 프레임	데이터 정상성			학습 적합성				해부학적 완전성		
		정상 프레임	학습 노이즈 프레임	평가 결과	양성 프레임	음성 프레임	LUI 프레임	평가 결과	필수 요소	선택 요소	평가 결과
1	25,791	24,071	1,720	0.93	40	24,027	1,724	0.93	10	1	0.95
2	9,898	6,356	3,542	0.64	6	1,225	8,667	0.12	4	1	0.38
3	19,626	16,581	3,045	0.84	134	16,447	3,045	0.84	8	0	0.76
4	16,568	15,796	772	0.95	74	15,705	789	0.95	10	0	0.95
5	19,622	18,859	763	0.96	10	18,849	763	0.96	6	0	0.57
6	38,090	32,325	5,765	0.85	24	32,301	5,765	0.85	10	0	0.95
7	19,080	18,943	137	0.99	8	18,933	139	0.99	8	0	0.76
8	17,768	17,586	182	0.99	19	17,551	198	0.99	10	0	0.95
9	26,806	25,357	1,449	0.95	19	25,337	1,450	0.95	10	0	0.95
10	30,070	29,934	136	1.00	96	29,804	170	0.99	5	0	0.48
11	29,684	28,710	974	0.97	2	28,704	978	0.97	9	0	0.86
Total	253,003	234,518	18,485	0.92	432	228,883	5,203	0.87	90	2	0.78
* 해부학적 완전성을 평가하기 위한 학습 목적 선택적 요인(Selective Factor)은 0.95로 설정함											

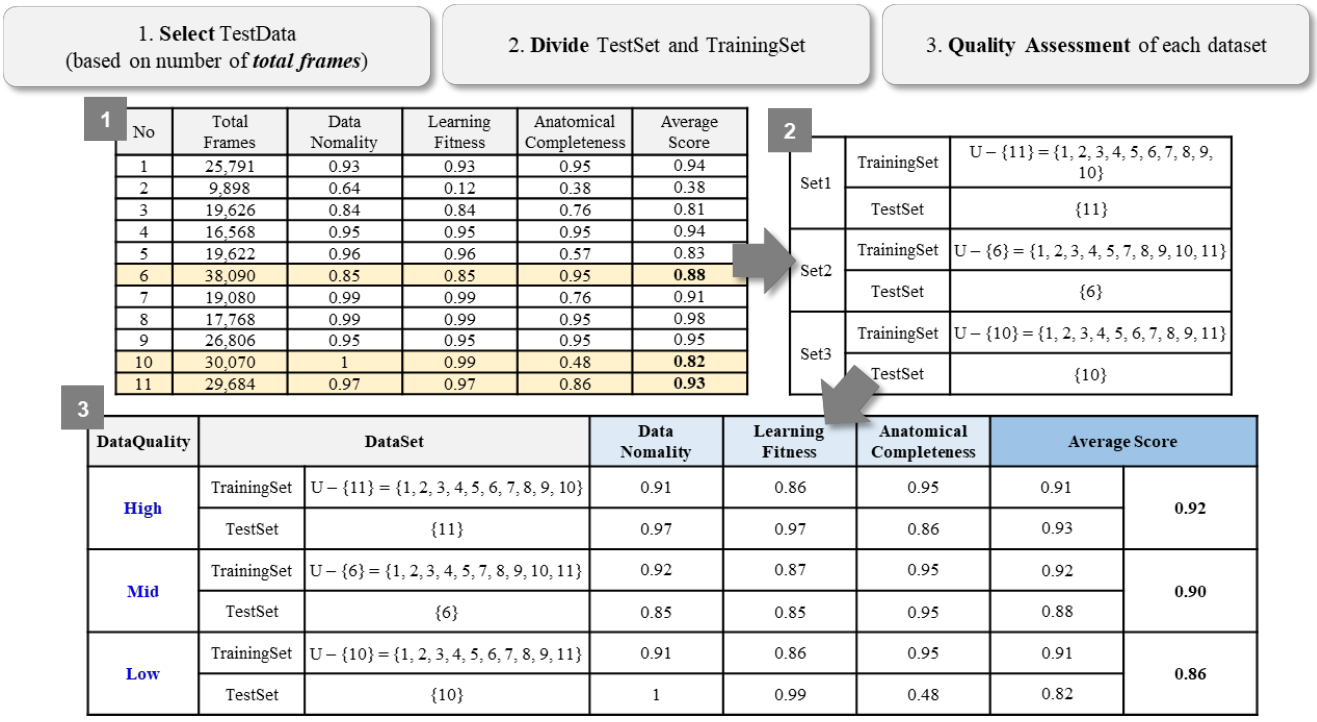


그림 5. 품질 별 학습 데이터 셋 생성 방법(High, Mid, Low)

다음은 학습 네트워크에 대한 설명이다. 본 논문에서는 그림 6과 같이 VGG-Net 구조를 기반으로 한 네트워크를 설계하였다. 그림에서 알 수 있듯이, 총 7개의 레이어(Layer)로 구성되어 있으며, 세부적으로는 10개의 컨볼루션(Convolution) 레이어와 5개의 풀링(Pooling) 레이어, 3개의 완전 연결(Fully-connected) 레이어로 구성되어 있다. 그러나 이와 같은 네트워크 구성만으로는 학습이 이루어지지 않는다. 의료 데이터의 특성상, 양성 데이터(Positive Data)가 음성 데이터(Negative Data)에 비해 굉장히 적은 비균형 데이터 문제(Problem of Imbalanced Data)가 발생하기 때문이다. 본 연구에서는 이와 같은 문제를 해결하기 위해, 양성 데이터는 합성을 통해 증대하고, 음성 데이터는 무작위 추출을 이용하여 축소시켜 균형 데이터로 만들어주는 과정을 거쳤다. 구체적으로 데이터 합성은 입력 이미지(100 X 100)를 90°, 180°, 270°로 회전시켜, 원래 이미지 개수의 4배에 해당되는 데이터를 생성하였다. 이와 같이 생성된 균등 데이터 셋은 입력 레이어에 들어가서 학습되며, 학습 시 배치 크기(Batch Size)는 20, 에폭(Epoch) 횟수는 50번으로 설정하였다

학습을 진행한 결과 그림 7, 그림 8과 같은 결과가 도출되었다. 먼저, 그림 7은 학습 및 평가용 데이터에 대한 정확도를 나타낸다. 그림 7의 상단과 같이 에폭이 증가함에 따라 학습 정확도가 1에 가깝게 수렴하는 것을 확인할 수 있다. 그러나 품질 별로 비교해보았을 때, 크게 차이가 나는 것은 확인할 수 없었다. 다음은 평가용 데이터에 대한 결과이다. 그림 7의 하단은 평가용 데이터에 대한 정확도이다. 그림에서 확인할 수 있듯이, 고품질의 데이터 셋이 다른 품질에 비해 보다 높은 정확도를 보이는 것을 확인할 수 있다. 그림 8은 평가용 데이터에 대해 민감도와 특이도를 추가적으로 분석한 결과이다. 마찬가지로, 품질이 좋은 데이터 셋에 대해 보다 높은 성능이 측정된 것을 확인할 수 있다.

결과적으로 고품질의 데이터에 대해서는 정확도 0.96, 민감도 0.93, 특이도 1.00을 보였고, 중품질의 데이터의 경우 정확도 0.94, 민감도 0.92, 특이도 0.96를 보였다. 마지막으로 저품질의 데이터에 대해서는 정확도 0.87, 민감도 0.78, 특이도 0.97로 수렴하는 것을 확인할 수 있었다.

5. 결론

본 논문에서는 캡슐내시경 영상 데이터 셋의 품질을 평가하기 위한 매트릭을 제안하였으며, 이를 기반으로 학습 네트워크의 성능에 대한 분석 결과를 제시하였다. 분석 결과, 평가용 데이터 셋에 대해 고품질의 경우 0.96, 중품질의 경우 0.94, 저품질의 경우 0.87의 정확도를 확인할 수 있었다. 이와 같은 결과는 고품질의 데이터 셋이 학습 성능에 미치는 영향이 크다는 것을 확인할 수 있게 한 것으로, 데이터 셋의 품질에 대한 중요성을 나타내었다. 향후에는 제안하는 품질 평가 매트릭을 기반으로 보다 나은 양질의 데이터 셋을 보강할 수 있게 하고, 위장관 교차점 분류기의 성능을 고도화할 예정이다.

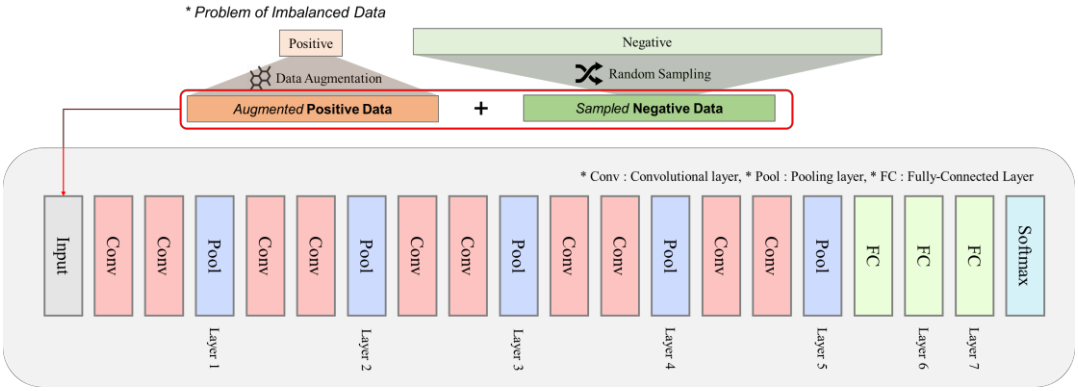


그림 6. 위장관 교차점 인식을 위한 네트워크 구조

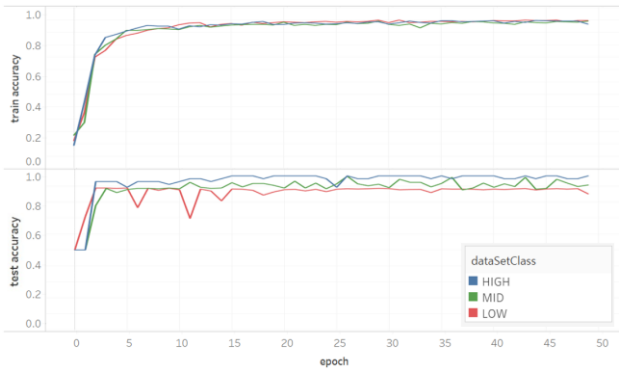


그림 7. 학습 및 평가용 데이터에 대한 정확도 분석

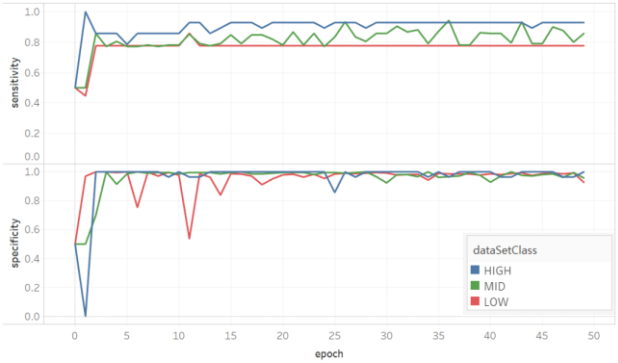


그림 8. 평가용 데이터에 대한 민감도 및 특이도 분석

참고 문헌

[1] Nahm, Meredith. "Data quality in clinical research." Clinical research informatics. Springer, London, 2012. 175-201.

[2] Gulshan, Varun, et al. "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs." Jama 316.22 (2016): 2402-2410.

[3] Reginatto, M., M. Anton, and C. Elster. "Assessment of CT image quality using a Bayesian approach." Metrologia 54.4 (2017): S74.

[4] Coyner, Aaron S., et al. "Deep learning for image quality assessment of fundus images in retinopathy of prematurity." blood vessels 1 (2018): 4.

[5] Xu, Kele, et al. "Full-Reference Image Quality Assessment-Based B-Mode Ultrasound Image Similarity Measurement." Medical Physics. Vol. 44. No. 6. 111 RIVER ST, HOBOKEN 07030-5774, NJ USA: WILEY, 2017.

[6] Yuan, Yixuan, and Max Q-H. Meng. "Deep learning for polyp recognition in wireless capsule endoscopy images." Medical Physics, vol.44, No.4 pp.1379-1389, 2017

[7] Li, Panpeng, et al. "Convolutional neural networks for intestinal hemorrhage detection in wireless capsule endoscopy images."2017 IEEE International Conference on Multimedia and Expo (ICME), 2017.

[8] Jia, Xiao, and Max Q-H. Meng. "A deep convolutional neural network for bleeding detection in Wireless Capsule Endoscopy images." 2016 IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society (EMBC), 2016.

[9] Jia, Xiao, and Max Q-H. Meng. "Gastrointestinal bleeding detection in wireless capsule endoscopy images using handcrafted and CNN features." 2017 39th Annual International Conference of the IEEE, Engineering in Medicine and Biology Society (EMBC), 2017

[10] Chen, Honghan, et al. "Automatic content understanding with cascaded spatial-temporal deep framework for capsule endoscopy videos." Neurocomputing, vol.229, pp.77-87, 2017.

[11] Lars Aabakken, et al, "Minimal standard terminology for gastrointestinal endoscopy-MST 3.0" World Endoscopy Organization, 2016.

[12] L. Y. Korman, et al, "Capsule endoscopy structured terminology (CEST): proposal of a standardized and structured terminology for reporting capsule endoscopy procedures." Endoscopy 37, pp.951-959, 2005.

스마트 컨트랙트 및 DApp 신뢰성 향상을 위한 단계별 체계적인 테스트 기법

민경식 ○[1], 이정원[2], 이병정[1]
[1]서울시립대학교 컴퓨터과학과
{ksmin1710, bjlee}@uos.ac.kr
[2]아주대학교 전자공학과
jungwony@ajou.ac.kr

A Systematic Test with Phases to Improve Reliability of Smart Contract and DApp

Kyeongsic Min ○[1], Jung-Won Lee[2], Byungjeong Lee[1]
[1]Department of Computer Science and Engineering, University of Seoul.
[2]Department of Electrical and Computer Engineering, Ajou University.

요 약

스마트 컨트랙트는 블록체인 기술을 기반으로 자동 계약 실행, 투명한 거래 등의 특징을 가지고 있다. 스마트 컨트랙트 기능을 제공하는 대표적인 프로젝트인 이더리움은 스마트 컨트랙트를 보다 편하게 개발할 수 있는 솔리디티 언어(Solidity) 및 이더리움 가상 머신을 제공한다. 또한, 스마트 컨트랙트를 활용하여 새로운 블록체인을 개발할 필요 없이 DApp을 만들 수 있게 되면서 많은 관심을 받고 있다. 하지만 현재 스마트 컨트랙트 기술은 코드 업데이트가 불가능한 특성을 가지고 있다. 따라서 컨트랙트의 배포 이후 결함이 발견될 경우, 사소한 결함이라도 이를 수정하기 위해 많은 비용이 필요하다는 단점이 있다. 따라서, 스마트 컨트랙트 개발은 기존 어플리케이션과는 다른 소프트웨어 공학적 접근이 필요하다. 즉, 개발된 컨트랙트를 배포하기 이전에 충분한 테스트 혹은 분석하여 결함이 없음을 확인해야 할 필요가 있다. 이를 위해 본 논문에서는 동적 테스팅 방법을 통해 스마트 컨트랙트 및 DApp의 정확성을 검증하고, 결함을 발견하여 신뢰성을 향상하는 테스트 기법을 제안한다. 이를 위해 각 테스트 단계에서 필요한 동적 모델을 정의하고 기존 테스팅 방법인 속성 기반 및 모델 기반 테스팅을 단계에 맞게 적용한다.

1. 서 론

블록체인(Blockchain)은 기존의 중앙 서버 방식을 탈피하여 제 3자의 개입 없이도 투명한 거래가 가능하다는 특징이 있다. 블록체인이 발전하며 주목을 받은 또 다른 개념은 스마트 컨트랙트(Smart Contract)이다[1]. 이 기능을 이용해 개발자들은 새로운 블록체인을 개발할 필요 없이 DApp(Decentralized Application)을 만들 수 있다. 하지만 BOS(Blockchain-oriented Software) 개발은 기존의 어플리케이션 개발과는 차이가 있기 때문에 새로운 소프트웨어 공학적 접근이 필요하다[2]. 또한, 신뢰성과 보안성을 보장해야 하는 블록체인의 특성 상 보다 많은 결함을 발견하기 위해 철저한 테스팅이 필요하다[3]. 특히, 현재 스마트 컨트랙트 개발의 경우 첫 배포 이후 코드 업데이트가 불가능하고, 스마트 컨트랙트의 결함을 수정하기 위해서는 많은 작업 혹은 비용이 요구되며, 결함으로 초래된 문제를 되돌리는 것이 불가능할 수도 있다. 즉, 스마트 컨트랙트는 블록체인보다 개발이 용이하지만, 숙련되지 않은 개발자가 개발할 경우 영구적인 결함이 생길 수 있다. 따라서 이러한 결함들을 조기에 발견하기 위해서는 초기에 정확한 테스트 케이스를 개발하고, 개발된 소프트웨어를 정확하게 테스트하는 것이 중요하다. 하지만 테스트 케이스 개발 및 테스트 과정은 테스터의 지식과 경험에 좌우되어 신뢰성을 보장하기 힘들다. 모델 기반 테스트는 이러한 문제점을 보완하고자 테스트 설계를 효율적으로 수행하는 방법으로서 소개되었다. 또한, 속성 기반 테스팅을 이용하면 테스트 케이스 설계 없이도 단위 테스트 단계에서 대상 모듈의 완전성을 확인함으로써 컨트랙트 배포 전에 더 많은 결함을 발견해낼 수 있다.

본 논문에서는 모델 기반 테스트 및 속성 기반 테스트를 활용하여 스마트 컨트랙트 및 DApp의 신뢰성 향상을 위한 단계별 체계적인 테스트 기법을 제안한다.

2. 스마트 컨트랙트 및 DApp 단계별 테스트 기법

스마트 컨트랙트 및 DApp은 중앙 서버를 사용하는 방식과는 다소 차이가 있기에 테스트를 하는 과정에서 DApp 클라이언트와 스마트 컨트랙트들 간 상호작용의

정확성을 검증할 필요가 있다. 또한, 스마트 컨트랙트는 일반적인 결함뿐만 아니라 TOD와 같은 블록체인의 특성에 의해 야기된 결함에 대한 고려가 필요하다. 본 논문에서는 블록체인 및 DApp의 특성을 고려함과 동시에 표 1과 같이 스마트 컨트랙트의 테스트 단계를 단위 테스트, 통합 테스트 그리고 시스템 테스트로 나누며, 각 테스트 단계에 필요한 모델을 제안한다.

표 1. 테스트 단계 구분

테스트 단계	테스트 대상	필요 모델	테스트 기법
단위 테스트	단일 스마트 컨트랙트	-	속성 기반 테스트
통합 테스트	스마트 컨트랙트	시퀀스 다이어그램	모델 기반 테스트
시스템 테스트	스마트 컨트랙트 & DApp	유즈케이스 액티비티 다이어그램	모델 기반 테스트

2.1 단위 테스트 단계

단위 테스트 단계에서는 속성 기반 테스트를 사용하여 소스코드를 메소드 단위로 검증한다. 이는 기존 테스트 케이스를 이용한 단위 테스트를 사용할 경우 테스트 케이스 개발이 테스터의 지식과 경험에 좌우되어 신뢰성을 보장하기 힘들기 때문이며, 모델 기반 테스트를 사용할 경우 단위 테스트를 위해 너무 많은 모델을 생성해야 하는 단점이 존재하기 때문이다. 속성 기반 테스팅은 많은 수의 테스트 케이스를 임의로 생성하기에 개발자가 불완전한 테스트 케이스를 단위 테스트에 이용하여 결함을 발견하지 못하는 등의 단점을 보완할 수 있다.

2.2 통합 테스트 단계

통합 테스트 단계에서는 확장된 시퀀스 다이어그램이 모델 기반 테스트를 위해 사용된다. 확장된 시퀀스 다이어그램은 각 유즈케이스를 표현하며, UML의 확장 매커니즘인 스테레오 타입을 이용하여 표현된 블록체인의 계정과 블록 인스턴스 등을 포함한다. 해당 단계에서는 테스트 커버리지 기준으로 술어 전 범위 기준(Full Predicate Coverage Criterion)를 사용하여 테스트 케이스를 생성한다. 이를 통해 최소한의 테스트 케이스로 정확도 높은 테스트를 수행할 수 있도록 한다.

2.3 시스템 테스트 단계

시스템 테스트 단계의 모델로 사용되는 유즈케이스는 시스템이 사용자에게 제공해야 하는 상위 수준 기능성을 나타내며, 이를 통해 시스템의 테스트 요구사항을 파악할 수 있다. 하지만, 각각의 요구사항은 독립적이지 않고, 다른 요구사항에 의존성을 가질 수 있다. 따라서, 시스템 테스트를 위해서는 유즈케이스들 간의 의존성을 고려해야 한다. 본 연구에서는 유즈케이스를 액티비티 다이어그램으로 표현한다. 또한, 유즈케이스에 의해 수정된 컨트랙트의 상태 배포 및 동기화에 필요한 최소 블록 검증의 수를 표기한다. 이를 기반으로 시스템 테스트 단계에서 유즈케이스들 간의 의존도와 필요한 블록 검증을 고려하여 테스트 케이스를 생성하고, DApp과 스마트 컨트랙트의 상호작용이 올바르게 작동하는지 검증할 수 있다.

3. 결론 및 향후 계획

본 논문은 단위 테스트 단계에서 속성 기반 테스트를, 통합 테스트, 시스템 테스트 단계에서는 모델 기반 테스트를 통해 스마트 컨트랙트 및 DApp의 신뢰성 및 정확성을 체계적으로 테스트하는 기법을 제안하였다. 체계적인 각 테스트 단계를 거쳐 스마트 컨트랙트 정적 취약점 분석 도구들로는 발견할 수 없는 일반적인 소프트웨어 결함을 발견할 수 있으며, 개발된 소프트웨어의 요구사항 충족 여부를 확인할 수 있다. 향후 연구로는 스마트 컨트랙트 정적 취약점 분석 도구들을 이용하여, 해커들에 의해 공격당할 수 있는 취약점들을 추가적으로 식별할 수 있도록 연구할 계획이다. 또한, 본 논문에서 제안한 테스트 기법을 자동화하여 병렬 소프트웨어 테스트가 가능하도록 연구 및 구현하여 더욱 신뢰성 있는 스마트 컨트랙트 및 DApp 개발이 가능하도록 지원할 계획이다.

4. Acknowledgement

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 차세대정보·컴퓨팅기술개발사업(NRF-2014M3C4A7030504)과 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 중견연구자지원사업(No.2017R1A2B4009937)의 지원을 받아 수행된 연구임.

참고문헌

[1] N. Szabo, “Smart Contracts: Building Blocks for Digital Markets”, http://www.fon.hum.uva.nl, 1996.

[2] S. Porru, A. Pinna, M. Marchesi, R. Tonelli, “Blockchain-oriented Software Engineering: Challenges and New Directions”, in Proc. of the 39th International Conference on Software Engineering Companion, pp. 169-171, 2017.

[3] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, R. Hierons, “Smart Contracts Vulnerabilities: A Call for Blockchain Software Engineering?”, in Proc. of International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pp.19-25, 2018.

차량 수준 재난요인분석 및 위험심사활동의 취약점과 실용적 가이드라인 연구

정효은○, 민상윤
한국과학기술원 전산학부
hejung@kaist.ac.kr, symin@kaist.ac.kr

Research on the weaknesses of vehicle-level hazard analysis and risk assessment actives and a practical guideline

Hyoeun Jung ○, Sangyun Min
Korea Advanced Institute of Science and Technology

요 약

자동차에 탑재되는 전자 시스템이 증가하고 있으며 시스템 결함으로 인한 사고 사례가 발생하면서 자동차 전자시스템의 안전성 확보에 대한 필요성이 대두되었다. 자동차를 생산하는 OEM은 자동차 기능안전 국제표준규격인 ISO26262에서 제시하는 HARA(Hazard Analysis Risk Assessment) 활동을 통해 시스템에서 발생 가능한 위험한 사건들을 발견하고 안전대책을 마련하고 있다. 또한 ISO26262 2판에서는 3.5톤을 초과하는 대형상용차에도 안전 활동의 필요성을 요구하고 있다. 하지만 HARA활동을 위한 과거 차량 데이터의 부족과 대형상용차의 특수성이 포함된 차량 수준의 체계적인 Hazard Analysis Risk Assessment 가이드라인의 부재로, HARA활동을 수행함에도 불구하고 예상하지 못한 재난 상황이 발생하는 문제점이 존재한다. 따라서 본 논문에서는 대형상용차를 포함한 차량 수준의 HARA 활동을 실용적이고 체계적으로 수행할 수 있도록 Hazard Analysis Risk Assessment 가이드라인을 제안한다. 본 방법을 통하여 차량 수준에서 다양한 관점으로 발생할 수 있는 재난요인을 체계적으로 고려하게 하고, 과거에 누락되어 있는 재난 요인 분석 및 위험 심사 활동을 보완한 기능안전목표를 설정함으로써, 차량의 신뢰성 있는 기능 안전 대책을 제시하는 것을 목표로 한다.

1. 서 론

자동차기술의 발전으로 운전자의 개입이 감소하고 자동차에 탑재되는 전자시스템의 의존도가 높아지고 있다. 전자시스템의 결함이 운전자의 생명과 밀접하게 연관되기에, 완성차(OEM)는 차량 수준에서 시스템이 발생 가능한 모든 위험한 사건에 대해 안전대책을 수립할 의무가 있다. 또한 대형상용차의 사고가 증가하면서 대형상용차에 ADAS(능동형 운전자 보조시스템, Advanced Driver Assist System)기술 적용 및 시스템 안전성 확보의 필요성이 증가하고 있다.

국제기구 ISO에서는 자동차 기능 안전 국제표준 ISO26262[1][2]을 제정함으로써 자동차에 탑재되는 시스템이 안전요구사항 준수를 통해 안전을 확보할 수 있게 하였다. ISO26262 요구사항 중 Part. 3 Concept Phase의 HARA(Hazard Analysis and Risk Assessment)활동을 통해 시스템의 재난요인을 도출하고, 재난요인을 대처하기 위한 안전목표를 도출한다. 하지만 HARA 활동에서 차량운용상황은 차량수준에서 재난요인을 도출하고, 재난요인이 차에 미치는 영향을 분석하는데 필요한 중요한 요소지만, 차량운용상황이 DB화 되어있지 않고, 차량 수준에서 체계적인 HARA활동 가이드라인이 부재한다. 이로 인해 재난요인 추출 시 차량운용상황이 일부 요소만 고려되고, HARA활동을 수행함에도 불구하고 예상하지 못한 재난요인이 발생하여, HARA활동 결과의 신뢰성이 감소하는 문제점이 존재한다.

또한 ISO26262 1판[1]에서는 3.5톤을 초과하는 대형상용차는 적용 대상이 아니다. 따라서 기존 대형상용차시스템은 ISO26262를 적용하지 않거나, 대형상용차의 특수성을 반영하지 않은 안전 활동을 수행하여 시스템의 안전 신뢰성을 보장할 수 없는 문제점이 존재하였다. 하지만 2018년 12월 개정된 ISO26262 2판[2]에서는 3.5톤을 초과하는 대형상용차가 적용 대상으로 추가됨으로써, 대형상용차에도 신뢰성 있는 안전 활동을 요구하고 있다.

이를 해결하기 위해 본 논문에서는 대형상용차의 특성이

포함된 차량운용상황을 데이터화 하고, 재난 분석 활동을 체계적으로 수행할 수 있는 현실적인 차량 수준의 Hazard Analysis Risk Assessment 가이드라인을 제안한다. 본 가이드라인은 대형상용차의 특수성을 반영한 위험심사방법을 포함한다. 본 방법을 통하여 다양한 차량운용상황에서 발생할 수 있는 재난요인을 포괄적이고 체계적으로 고려하고, 과거에 누락되어 있는 재난요인분석 및 위험심사활동을 보완한 기능안전목표를 설정함으로써, 대형상용차를 포함한 차량의 신뢰성 있는 기능안전목표를 제시하는 것을 목표로 한다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 HARA 표준과 한계점에 대해 언급한다. 3장에서는 차량운용상황 데이터를 분석하여 모델화하고, 4장에서는 상용차 특성을 보완한 Severity 결정 방법을 소개한다. 5장에서는 3, 4장의 방법을 적용하여 체계적으로 차량운용상황을 고려하게 하고 대형상용차의 특성을 반영한 Hazard Analysis Risk Assessment 가이드라인을 제안한다. 6장에서는 가이드라인을 대형상용차에 적용해보고 평가함으로써 가이드라인의 실현 가능성을 보여준다. 마지막 7장에서는 결론 및 향후 연구에 대해 설명한다.

2. J2980 표준 및 한계점

SAE J2980 – Considerations for ISO 26262 ASIL Hazard Classification[3]은 ISO26262에서 요구하고 있는 HARA활동에 대한 가이드라인을 제안한다. J2980은 HARA에 사용되는 차량운용상황조건을 그림 1과 같이 제시하고 있지만, 42개의 차량운용상황은 차량에서 실제 발생 가능한 다양한 차량운용상황을 모두 고려하지 못하며, 대형상용차와 같은 특수 차량의 특성을 포함하지 않고 있다. 엔지니어가 누락할 수 있는 더 다양하고 상세한 차량운용상황에 대한 고려가 필요하다. 또한 차량운용상황은 동시에 발생할 수 있지만, J2980에서는 조합 가능한 차량운용상황에 대한 가이드가 부재하다.

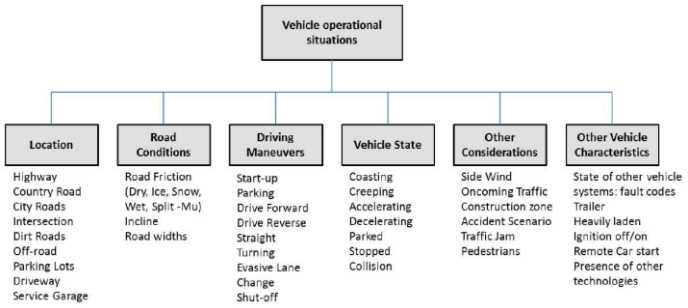


그림 1. J2980 잠재적 차량운용상황

HARA는 재난요인이 차량 수준에 미치는 위험요소 Severity, Exposure, Controllability를 평가하는 활동을 포함한다. J2980에서 Severity는 표 1과 같이 시스템의 재난요인으로 인해 사고가 발생할 경우, 차량 속도에 따른 인체 상해 정도를 반영하여 평가하고 있다. 하지만 이 기준은 3.5톤 초과의 대형상용차의 특성이 반영되어 있지 않다. 3.5톤 이하의 작은 차와 대형상용차가 같은 차 속 조건에서 사고가 발생할 경우, 대형상용차의 조건이 더 심각 한 인체 상해 정도를 입힐 수 있기에 표 1은 대형상용차에 동일하게 적용할 수 없는 한계점이 존재한다.

표 1. J2980 차 속에 따른 심각도 평가 기준

Collision Type	Range For	S0	S1	S2	S3
Front	Minimum speed		>4-10kph	>20-50kph	>40-65kph
	Maximum speed	<4-10kph	<20-50kph	<=40-65kph	
Back	Minimum speed		>4-10kph	>20-50kph	>40-60kph
	Maximum speed	<4-10kph	<20-50kph	<=40-60kph	
Side	Minimum speed		>2-10kph	>=8-30kph	>16-40kph
	Maximum speed	<4-10kph	<8-30kph	<16-40kph	

3. 차량운용상황 분석 및 모델화

3.1. 차량운용상황 분석

HARA활동 수행 시 시스템이 차량 수준에서 영향 받을 수 있는 다양한 동작 환경들을 파악하기 위해, 차량 수준에서 발생할 수 있는 차량운용상황을 추출하여 나열하였다. 추출한 차량운용상황은 223개로, FMEA에서 시스템 고장에 영향을 주는 요소와 대형상용차의 특수성을 반영하였다.

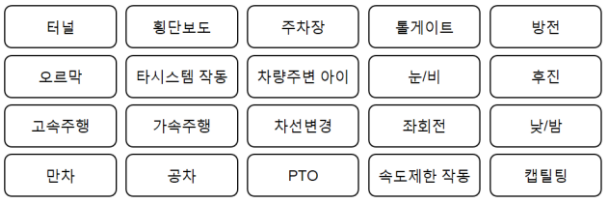


그림 2. 차량운용상황의 예

나열된 차량운용상황의 특성을 분석하였을 때 차량운용상황은 동시에, 조합적 결합으로 발생할 수 있다. 차량운용상황(S)은 동시에 발생 가능한 하위차량운용상황(S_i) 요소들로 구성되어 있다($S = S_1 \,||\, S_2 \,||\, \ldots \,||\, S_n$). 하위차량운용상황(Si) 는 Don't Care(0), Care(1)의 2개의 상태가 존재하며, 고려해야 할 차량운용상황이 총 n개 일 경우, 조합된 전체 차량운용상황은 2^n 개이다. 하지만 하위차량운용상황이 30개만 넘어도 10억 개가 넘는 차량운용상황 시나리오가 생성되어, 현실적으로 모든 요소가 조합된 차량운용상황을 확인할 수 없는 상태폭발 (State Explosion) 문제가 존재한다.

3.2. 차량운용상황 모델화

조합된 차량운용상황들의 상태폭발문제를 해결하기 위해 추상화 방법을 사용한다. 3장에서 추출한 223개의 차량운용상황 값들이 유사한 특성을 가지는 성질에 대해 그룹화하여 모델을 추출하였으며, 각 모델들을 다시 추상화여 메타-모델을 추출하고, 메타-모델을 다시 추상화한 메타-메타-모델을 그림 3과 같이 추출하였다. 하위차량운용상황 조합 시 추상화된 메타-메타-모델수준, 메타-모델수준, 모델 수준에서 차량운용상황 시나리오 표현이 가능할 경우, 해당 시나리오는 각 추상화된 모델 수준에서 조합한다. 또한 각

추상화된 모델 수준에서 시스템이 고려할 필요 없는 모델, 메타-모델, 메타-메타-모델을 제외할 수 있다. 이와 같은 방법 적용 시, 각 모델 안에 속한 하위차량운용상황의 개수가 m개라면, 조합된 차량운용상황이 전체 2^m만큼 축소된 효과를 볼 수 있다.

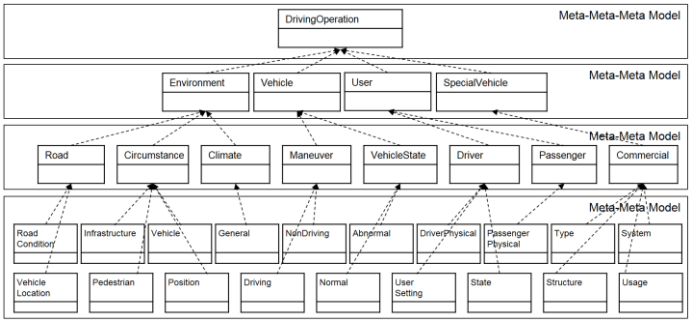


그림 3. 차량운용상황 모델들

4. 대형상용차특성을 반영한 Severity 결정 방법

기존 HARA의 Severity 결정 방법은 시스템의 재난요인으로 인해 사고가 발생했을 경우 차량 속도에 따른 인체 상해 정도를 반영하였다. 하지만 대형상용차의 특성상 대형 차량이 만 차 상태에서 고속 조건에 따른 충돌 시험에 대한 현실적 어려움이 존재하며, 이를 평가하기 위한 법적 규제도 미미한 상태이다. 현재 ANCAP에서 3톤이상 5톤이하의 차량에 대한 충돌 안전성에 대한 범위 측정 연구가 주도적으로 진행되고 있지만, 아직 초기단계이며 5톤을 초과하는 대형상용차를 대상으로 하지 않는다. 따라서 이를 보완하기 위해 대형상용차에서 사고가 발생할 경우 기존 Severity 결정 요소에 가중치를 반영하는 방법을 제안한다.

상용차는 용도에 따라 사람의 탑승 수, 차량의 무게 등 종류가 다양하기 때문에 유사한 특성에 따라 그룹화하였다. 상용차 그룹화 시 UNECE vehicle category Classification[4] 기준을 반영하였다(그림 4 참조). 시스템 적용 대상 차량이 M1/N1일 경우 가중치가 없으며, 시스템 적용 차량이 M2/M3/N2/N3일 경우 가중치 1을 인체 상해 등급에 추가하여 Severity를 재선정한다.

Category		description
M		used for the carriage of passengers
	M1	no more than eight seats in addition to the driver seat (mainly, cars)
	M2	more than eight seats in addition to the driver seat &having a maximum mass not exceeding 5 tonnes
	M3	more than eight seats in addition to the driver seat &having a maximum mass exceeding 5 tonnes
N		used for the carriage of goods (trucks)
	N1	not exceeding 3.5 tonnes.
	N2	having a maximum mass exceeding 3.5 tonnes but not exceeding 12 tonnes
	N3	having a maximum mass exceeding 12 tonnes

	구분	Severity 등급			
		S0	S1	S2	S3
AS IS	AIS*인체 상해등급	0	1,2	3,4	5
TO BE	AIS*인체 상해등급+ weighting factor	0	1,2	3,4	5

*AIS (Abbreviated Injury Scale) : 국제적 외상 평가 기준

그림 4. 상용차그룹카테고리 및 인체 상해 등급과 가중치에 따른 Severity 결정 표

5. Hazard Analysis Risk Assessment 가이드라인

본 논문에서 제안하는 Hazard Analysis Risk Assessment 가이드라인은 HARA를 수행하는 엔지니어가 현실적인 수준에서 다양한 차량운용상황 시나리오를 고려하게 하면서, HARA활동을 포괄적이고 체계적으로 수행할 방안을 제공한다.

첫 번째 단계로, 엔지니어는 대상 시스템이 영향을 받는 차량운용상황과 영향을 받지 않는 차량운용상황을 메타-메타모델, 메타-모델, 모델 수준에서 선택한다. 두 번째 단계로, 엔지니어가 시스템에 영향을 주는 하위차량운용상황들의 조합을 선택한다. 하위차량운용상황은 유사한 차량운용상황 값들을 대표하는 모델/메타-모델/메타-메타모델 수준에서 선택할 수 있다. 이때 선택된 하위차량운용상황과 조합될 수 없는 하위차량운용상황은 선택할 수 없게 하여 선택한 하위차량운용상황과 조합 가능한 가이드라인을 제시한다(예: 엔지니어가 고속도로를 선택할 경우 고속도로에서 발생할 수 없는 횡단보도, 주변외부환경아이, 정비소, 과속방지턱과 같은 하위차량운용상황은 비활성화). 엔지니어는 해당 가이드라인을 따라 조합 가능한 하위차량운용상황들을 선택하여 조합된 차량운용상황 시나리오를 도출한다. 추가로 엔지니어가 선택한 하위차량운용상황들(S_{select})={$S_{select1}$,$S_{select2}$,..,$S_{selecti}$}과 특정 하위차량운용상황들(S_{combi}) 에 대해 모든 조합된 시나리오를 보고 싶을 경우, 현실적으로 조합가능하도록 15개에 한하여 조합을 원하는 하위차량운용상황(S_{combi})={S_{combi1}, S_{combi2}, …,S_{combij}| j≤15}을 선택하게 하여 조합된 차량운용상황시나리오(S_{v1})를 만들어준다. 이때 조합된 차량운용상황 시나리오의 수(n(S_{v1}))는 $i*2^j$이다. 세 번째 단계로, 엔지니어는 두 번째 단계를 반복하며 조합된 차량운용상황들을 도출하여 최종 차량운용상황(S)={S_{v1}, S_{v2}, …,S_{vn}}을 도출한다. 네번째 단계로, 최종 도출된 차량운용상황(S)들이 첫 번째 단계에서 선정한 모델을 한 번도 고려하지 않은 경우, 누락 되어 있는 모델에 대응하는 하위차량운용상황들에 대해 재검토를 진행한다. 이 단계를 통해 엔지니어가 선택한 차량운용상황들이 차량 수준에서 발생 가능한 차량운용상황 시나리오들을 누락하지 않도록 하여, 범위의 포괄성을 확보한다. 마지막으로 도출한 차량운용상황(S)을 통해 시스템의 Malfunction이 차량운용상황(S)에 적용했을 때 발생하는 재난요인을 도출하고, 재난요인이 가지고 있는 위험성을 평가하여 최종적으로 ISO26262 안전목표 (ASIL, Automotive Safety Integrity Level)를 도출한다.

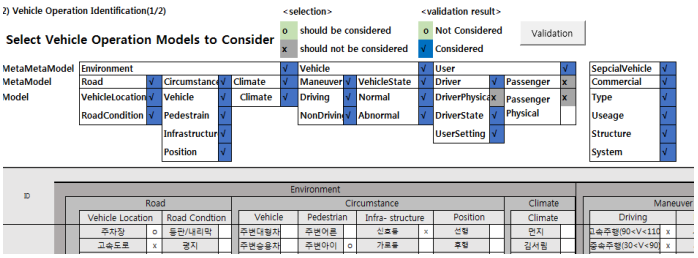


그림 5. Hazard Analysis Risk Assessment 가이드라인

6. 적용 및 평가

Hazard Analysis Risk Assessment 가이드라인을 대형상용차에 탑재되는 시스템에 적용하고, HARA활동 결과를 비교 분석하였다. 대상은 대형상용차의 원격제어시스템으로, 차량의 전원 및 시동을 리모컨을 통해 원격으로 제어하여

운전자의 편리성을 향상시키는 시스템이다. 시스템 적용 대상 차량은 14톤 트럭이다. 원격시동제어시스템 정의를 수행 후, HAZOP을 통해 Malfunction을 식별하였다. 제시한 Hazard Analysis Risk Assessment 가이드라인을 적용하여 HARA를 수행하여 최종 56개의 재난요인과 안전목표(ASIL) 등급 C를 도출하였다.

Hazard ID	Malfunction		Vehicle Operation		Vehicle Hazards	Consider Risk Assess-ment	RISK						ASIL	Comments
	Malfunction ID	Description	Vehicle-Operation ID	Description			E	Reason for E	S	Reason for S	C	Reason for C		
HZ184	MF010	의도치않은 시동 OFF	VO0016	시가지;젖은도로;주변아이;비;종속주행(30<V<90);좌우회전(유턴;만차;N4;덤프;	주행 중 의도치 않은 시동정지	Yes	3	비오는 조건에 시가지 환경	3	충돌상대속도 Max 50, 전/후면, 측면 충돌 기준이나 주변 보행자 가능	3	운전자가 차량제어가 매우 어려움	C	빙판길 차량 급 조향 조건에서의도치 않은 시동정지 발생 시 차체 제어 불가로 인한 Roll away발생 가능. 주변 보행자 존재가능

그림 6.원격시동제어시스템 Hazard Analysis Risk Assessment

기존 원격제어시스템에서 수행한 HARA활동 결과와 새로운 가이드라인을 적용한 결과를 비교하였을 때, 표 2를 참고하면 상대적으로 더 많은 재난요인과 다양한 차량운용상황을 고려하였고, 기존에서 발견하지 못한 누락된 차량운용상황을 발견하였다. 최종적으로 누락된 차량운용상황이 기존보다 높은 Risk를 발현하여 최종 안전목표(ASIL) 등급이 B에서 C로 상승하였다.

표 2. HARA활동결과 비교표

구분	기존 HARA활동결과	새로운 Guideline 적용 HARA활동 결과
도출된 Hazards 수	9건	56건
적용한 차량운용상황 데이터 수	데이터 : 40개, 카테고리 : 4개	데이터:213개, 모델 : 19개 -상용 차량 특성 반영
Risk Assessment Results	ASIL B - Hazard가 높은 Risk을 발현할 수 있는 차량운용상황 누락 - 상용 차량 특성 반영하고 있지 않음	ASIL C - 기존에 고려되지 않았던 차량운용상황을 Hazard에 반영하여 Risk 평가 수행 - 상용 차량 특성이 반영된 Hazard 추가 - 상용 차량 특성이 반영된 Severity Risk 측정 요소 반영

7. 결론

본 논문에서는 차량 수준에서 HARA활동을 포괄적이고 체계적으로 수행 할 수 있는 현실적인 Hazard Analysis Risk Assessment 가이드라인을 제시하였다. 대형상용차의 특성이 포함한 차량운용상황을 추출하고, 차량운용상황들의 특성을 분석하여 모델링하였다. 대형상용차 특성이 반영된 Severity 결정 방법을 제안하여 가이드라인에 반영하였다. 본 방법을 대형상용차 프로젝트에 적용하여 과거에 진행했던 결과와 비교하였다. 과거 대비 누락된 차량운용상황들을 발견하고, 고려하지 않았던 재난요인을 보완한 새로운 안전목표등급을 도출하였다.

완성 차를 다루는 OEM에서는 실 차 관점에서 발생 가능한 모든 위험한 사건들에 대해 식별하고 위험성을 평가하여 그에 따르는 안전대책을 마련해야 한다. 과거 엔지니어의 경험치에 의존적이었던 HARA활동이, 본 가이드라인을 통해 엔지니어의 경험치에 관계없이 더 많은 재난요인을 포괄적이며 체계적으로 고려하게하고, 과거에 누락되어 있는 위험요소를 식별하게 한다. 보완된 안전요구사항을 추출함으로써, 하위 부품수준에서도 충분한 안전장치가 구현되도록 통제할 수 있게 한다. 또한 대형상용차의 특성이 반영된 HARA활동을 수행하게 하고, 높은 기준의 위험심사활동을 통해 승용 대비 높은 위험을 지니고 있는 대형상용차에도 높은 수준의 안전을 확보하게 한다.

향후 본 가이드라인에 적용된 차량운용상황 데이터를 지속적으로 축적하고 확장한다. 특히 커넥티드 차량 및 자율주행자동차의 특수성을 추가하여, 변화하는 자동차환경에 유연하게 대응하고 미래자동차에도 신뢰성 있는 ISO26262활동이 가능하기를 기대한다. 향후에는 축적된 차량운용상황 데이터와 본 가이드라인을 활용한 HARA 활동결과 데이터를 활용하여, 학습 알고리즘으로 대상 시스템에 의미 있는 조합차량운용상황을 자동추출하는 방식을 통해 엔지니어의 개입없이 일관된 결과를 도출할 수 있는 기법을 연구하고자 한다.

8. 참고 문헌

[1] “ISO 26262:2011 Road Vehicles –Functional Safety”, International Organization for Standardization, 2011

[2] “ISO 26262:2018 Road Vehicles –Functional Safety”, International Organization for Standardization, 20118

[3] “SAE J2980 –Considerations for ISO 26262 ASIL Hazard Classification, Society of Automotive Engineers, 2015”

[4] “Special Resolution No. 1 concerning the common definitions of vehicle categories, masses and dimensions (S.R.1)”, ENECE, Vehicle regulations

[5] “Directive 2014/47/EU of the European Parliament and of the Council of 3 April 2014 on the technical roadside inspection of the roadworthiness of commercial vehicles circulating in the Union and repealing Directive 2000/30/EC Text with EEA relevance OJ L 127, 29.4.2014, p. 134–218 (BG, ES, CS, DA, DE, ET, EL, EN, FR, HR, IT, LV, LT, HU, MT, NL, PL, PT, RO, SK, SL, FI, SV)”, European Union Law

[6] P. Dhaussy, J. Roger and F. Boniol, "Reducing State Explosion with Context Modeling for Model-Checking," 2011 IEEE 13th International Symposium on High-Assurance Systems Engineering, Boca Raton, FL, 2011, pp. 130-137

[7] F. Heidari, P. Loucopoulos, F. Brazier and J. Barjis, "A Meta-Meta-Model for Seven Business Process Modeling Languages," 2013 IEEE 15th Conference on Business Informatics (CBI), Vienna, Austria, 2013, pp. 216-221.

[8] H. Kwon, R. Itabashi-Campbell and K. McLaughlin, "ISO26262 application to electric steering development with a focus on Hazard Analysis," 2013 IEEE International Systems Conference (SysCon), Orlando, FL, 2013, pp. 655-661.

스마트 빌딩을 위한 b.IoT 솔루션과 휴먼센싱 기술

임헌정
삼성 전자
{hjlim99}@imtl.skku.ac.kr,

b.IoT Solution and Human Sensing technologies for the Smart Building

Lim Hunjung
Samsung Electronics

요 약

오늘날 전체 인구의 90%는 도시에 거주하며 대부분의 생활을 빌딩에서 지내고 있다. 빌딩 관리 시스템은 BAS에서 에너지 관리를 위한 BEMS을 거쳐 Smart Building으로 확대되고 있다. 빌딩내 모든 기기 관리와 거주자 편의 서비스 제공을 위해서는 IoT 기술과 휴먼센싱 기술 적용이 필수 적이다. 따라서 본 논문에서는 빌딩내 사물을 위한 building IoT 요구 사항을 정의 하고, 거주자를 위한 휴먼센싱 기술에 대하여 기술 하였다. 본 연구에서는 LIDAR센서에서 수집된 점군의 전처리를 통해 분석 가능한 상태로 만들고 군집화 알고리즘 적용 하여 휴먼센싱이 가능함을 보였다.

1. 서 론

오늘날 국내 인구의 90%는 도시에 거주하며 대부분의 생활을 빌딩에서 지내고 있다. 빌딩 관리 시스템은 BAS(Building Automation System)에서 에너지 관리를 위한 BEMS(Building Energy Management System)을 거쳐 거주자의 편의와 주변 인프라와의 통합을 목적으로 하는 Smart Building으로 확대되고 있다[1]. 스마트 빌딩은 전력, 냉난방 시스템, 지능형 영상보안, 주차관리 등 다양한 솔루션이 적용되어 관리자의 입장에서는 인력과 비용절감 효과를 가져오며, 사용자 입장에서는 안전성과 편의성이 증대되는 효과가 있다. 시장조사기관인 마켓앤마켓 (Markets and Markets)의 조사 결과에 의하면 스마트 빌딩 시장은 2021년 247억 3,000억 달러까지 시장 규모가 늘어날 것이라고 전망하고 있다. 이에 세계적 기업인 슈나이더 일렉트릭, 허니웰, 지멘스 등이 활발히 연구를 진행하고 있다. 최근에는 사물인터넷(IoT)기술을 더해 빌딩의 주요 설비에 IoT 센서를 적용해 빌딩 내 모든 상황을 모니터링하고 이를 기반으로 스스로 상태를 판단해 최적의 운영가이드를 제공하는 수준까지 도달하고 있다.

본 연구에서는 기 확보된 공조, 네트워크, 무선단말기 하드웨어 기술과 데이터 분석/AI의 소프트웨어 기술을 융합하여 스마트 빌딩의 기능 확대 방안과 휴먼센싱 기술 적용을 위한 최근 연구 결과에 대하여 다루고 있다.

2. b.IoT 솔루션

기존의 빌딩내 시스템과 에너지 관리를 위한 공조 설비의 관리 영역을 확대하고 빌딩내 모든 기기 및 장비 관리를 위해서는 IoT 기술의 적용이 필수 적이다. 따라서 본 building IoT 연구에서는 다음과 같은 요구사항을 정의 하고 기술 개발을 진행 하였다.

-빌딩 자동제어 솔루션 : 빌딩 내 주요 설비와 시스템을 통합 운영 기능

- 빌딩 기기간 무선 통신 : 기존 유선 방식 대비 관리 및 설치가 유리한 무선 통신 관리 기능

- 지능형 에너지 절감 : 각 센서 및 운영기기에서 수집되는 데이터 분석을 통한 제어 기능

- 모바일 관제 : 관제실과 모바일 기기간 기능 동기화를 통한 이상 운전 및 비상상황 알림 기능

그림 1 b.IoT 솔루션 UI

3. 휴먼 센신 기술

스마트 빌딩의 주요 목적인 거주자에 대한 편의와 안전성 제공을 위하여서는 거주자에 대한 속성, 상태, 위치, 동선 파악 등이 필요 하다. 이러한 휴먼 센싱을 위해서는 다음과 같은 기술이 적용 가능하다.

- 속성/상태 : 성별, 나이, 기분 등에 대한 파악으로 카메라 기반 DNN기술 적용이 가능하다.
- 위치/동선 : 현재위치, xyz좌표 등으로 AP신호를 이용한 삼각측량, LIDAR센서 등을 이용한 물체탐지 기술이 적용 가능하다.

본 장에서는 휴먼센싱 기술 중 위치/동선을 LIDAR센서를 이용하여 판단하는 기술에 대하여 다루고 있다.

3.1 LIDAR

LIDAR 센서는 빛을 이용하여 물체와의 거리를 측정하는 센서로써 빛의 비행시간(TOF)정보를 활용한다. 최근의 LIDAR는 센서를 1~64개까지 연결하여 한번에 어러개의 거리 정보(point)를 획득하여 POINTCLOUD (점군)를 생성하여 공간 구조를 재 구성 할 수 있다.

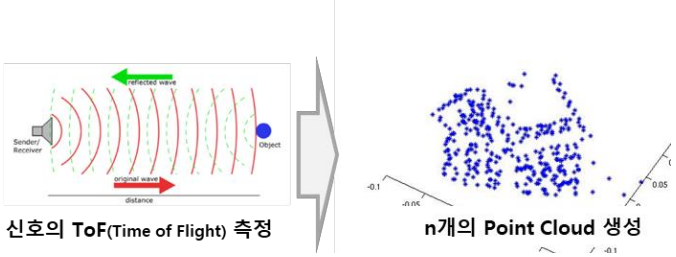


그림 2 LIDAR센서 원리 및 Point Cloud

3.2 데이터 전처리 및 배경 제거

생성된 점군들은 PCL(Pointcloud Library)을 활용하여 복셀화, 노이즈제거, 배경제거, 관심영역 설정 등 전처리를 통해 분석이 용이한 상태로 변경가능하다[2]. 전처리 작업 중 배경제거 및 관심영역 설정은 추후 물체 탐지 정확도에 큰 영향을 미치게 됨으로 신중하게 진행 되어야 한다. 본 연구에서는 분석 대상이 되는 공간을 로비나 강당처럼 사람만 존재 하는 자유공간(Free Space)과 사물실과 교실처럼 사람과 사물이 공존하는 복합공간(Dense Space)으로 나누어 공간 특성별로 배경 제거와 관심영역 설정을 따로 실시 하였다.

- Free Space : 관심 영역 설정 (PCL의 PasThrough 필터 이용)
- Dense Space : 배경 제거 (Octree구조체를 이용한 점군 탐색 및 제거 이용)

최종 결과물에 대하여 군집화 알고리즘인 DBSCAN을 적용하여 의미있는 물체 집합들로 나누어 사람을 탐지 하였다[3].

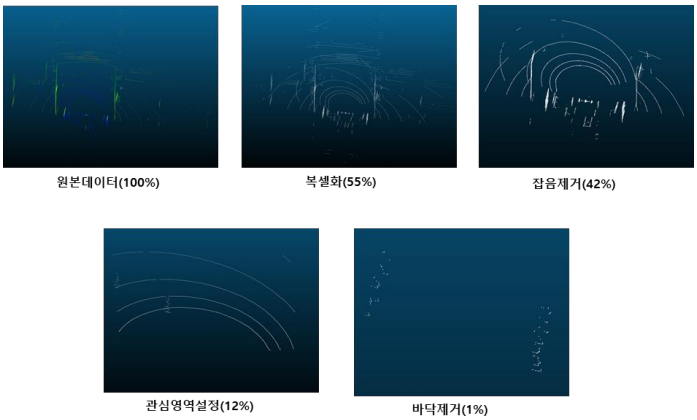


그림 3 Point Cloud 전처리(포인트 수 변화)

4. 결론 및 향후 연구

본 연구에서는 스마트 빌딩 구현을 위하여 필요한 b.IoT 솔루션과 휴먼센싱 기술에 대하여 살펴 보았다. 휴먼센싱 기술 중 거주자 위치 및 동선 파악을 위해서는 LIDAR센서가 활용 가능하다. 본 연구에서는 LIDAR센서를 통해 수집된 점군을 전처리를 통해 분석이 용이한 상태로 만들고 군집화 알고리즘을 통해 의미있는 물체별로 나누어 휴먼센이이 가능함을 보였다. 향후 연구를 통해 군집화 된 물체에 대해 분류 기술을 적용하여 책상과 사물, 파티션과 사물을 구분 하려 한다.

References

[1] 임헌정, 이동섭, 서성목, 조혜정. (2014). 에어컨 운영 데이터를 이용한 빌딩 에너지 모델 보정 시스템 설계. 한국통신학회 학술대회논문집, , 113-114.

[2] Radu Bogdan Rusu and Steve Cousins, “3D is here: Point Cloud Library (PCL)”, IEEE International Conference on Robotics and Automation (ICRA), May 9-13 2011

[3] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226-231.

다양한 신경세포 모델의 행동 양식 특징 분류를 통한 효과적인 코드 관리

이헌준 ○ 김장우
서울대학교 전기·정보공학부
hunjunlee7515@snu.ac.kr, jangwoo@snu.ac.kr

An Efficient Code Management with Classification of Diverse Neuron Model Based on Behavioral Characteristics

Hunjun Lee○ Jangwoo Kim
Department of Electrical and Computer Engineering, Seoul National University

요 약

최근 딥러닝에 기반한 알고리즘은 기존의 알고리즘에 비해 패턴인식이나 분류와 같은 복잡한 연산을 매우 간단하고 쉬운 구조로 해결한다. 그러나, 이러한 인공지능 알고리즘들은 생물학적 신경망이 나타내는 복잡한 연결에 착안해서 발전하였음에도, 실제 생명체 내 신경세포의 연산 방식과는 많은 차이를 보인다. 따라서 실제 신경세포의 연산 방식을 딥러닝 구조에 적용하면 기존 방식으로 해결할 수 없었던 다양한 문제를 해결할 수 있을 것이라 예상된다. 하지만 신경세포는 다양하고 복잡한 방식으로 동작하기 때문에 프로그래머는 각 모델에 따로 별도로 코드를 관리하는 어려움에 봉착한다. 본 연구에서는 이러한 어려움을 해결하기 위한 다양한 신경세포 모델을 행동 양식으로 나누어 코드 관리를 편하게 하는 방법을 연구한다.

1. 서 론

Artificial Neural Network(ANN)의 차세대 Neural Network로 제안되는 Spiking Neural Network(SNN)는 스파이크(Spike)라는 전기적 신호를 통해서 뉴런 연산을 수행하고, 뉴런 간 정보를 전달한다. 특히, SNN은 스파이크 개념을 네트워크에 적용함으로써 정보 사이의 시간적 개념을 연산 방식에 도입하였다. 이러한 특징으로 인해, SNN은 ANN을 통해 수행하기 어려웠던 다양한 연산들을 가능하게 할 새로운 Neural Network 중 하나로 주목을 받으며 활발하게 연구되고 있다.

SNN에서는 스파이크를 이용한 정보의 연산과 전달을 수행하기 위해 매우 복잡하고 다양한 모델들을 사용하여 신경세포의 연산을 설명한다. 하지만 스파이크 연산에 있어 최적화된 모델은 아직 알려지지 않았으며, 이 때문에 다양한 종류의 신경세포 모델을 통해서 신경망을 구성하는 여러 연구가 존재한다 [1][2][3].

신경세포 모델이 더욱 복잡해지면서 SNN의 연산 능력의 잠재력을 확인할 수 있었지만, 프로그래머는 다양한 신경세포 모델을 새롭게 프로그래밍 해야 한다는 어려움에 봉착하게 된다. 예를 들어, 하나의 SNN 안에서 다양한 신경세포 모델이 사용되거나, 다양한 신경세포 모델을 제공하는 NEST [4], pyNN [5], NEURON [6] 과 같은 시뮬레이터를 만드는 경우 프로그래머는 다양한 신경세포 모델을 구현하고, 해당 코드를 관리해야 한다. 또한, 새로운 신경세포 모델이 생길 때마다 기존의 모델을 활용하지 못하고, 새로운 모델을 구현해야 하는 노력이 요구된다. 따라서, 이를 효율적으로 구현하기 위한 새로운 기법의 프로그래밍 방법론이 필요하다.

본 연구에서는 여러 신경세포 모델을 정의할 수 있는 13가지 특징을 정의하고, 정의된 특징을 5가지 범주로 분류한다. 그 후, 분류된 특징을 조합하여, 현존하는 다양한 뉴런 모델들을 표현할 수 있음을 보이고, 해당 방법이 코드의 관리 및 확장성 측면에서 더 유리함을 보인다.

2. 배경지식

사람의 두뇌는 신경세포라는 기본 연산 단위가 방향성을 가지고 서로 연결되어 있는 구조이다. 각 신경세포는 다른 신경세포로부터 스파이크의 형태로 전달되는 정보를 받고, 받은 스파이크의 패턴에 따라 내부 막 전위를 변화시킨다. 이때, 변화된 막 전위가 특정 역치 값을 초과하면, 신경세포는 새로운 스파이크를 발생시킨다. 많은 신경과학자는 이러한 신경세포의 메커니즘을 수학적으로 표현하고자 연구해왔는데, 대표적인 모델로는 Hodgkin-Huxley 모델 [8](HH 모델), Leaky-Integrate and Fire 모델 [7](LIF 모델), Izhikevich 모델 [3] 등이 있다.

HH 모델은 현존하는 모델 중 신경세포의 동작 방식을 가장 유사하게 나타내는 모델이다. 다른 모델과 다르게 HH 모델은 막 전위 값의 변화에 영향을 미치는 이온들을 각각 다르게 모형화하고, 서로 다른 이온들이 어떻게 합쳐져서 막 전위에 영향을 미치는지 설명한다. 이렇게 복잡한 구조를 기반으로, HH 모델은 하나의 신경세포가 나타내는 다양한 생물학적

특징을 설명할 수 있다. 하지만 HH 모델은 너무 많은 연산 오버헤드를 발생시키기 때문에 많은 뉴런을 가지는 신경망을 모형화 하는 경우에는 비효율적이다.

HH 모델의 문제점을 해결하기 위해 LIF 모델 [7]과 Izhikevich 모델 [3] 등이 제안되었다. 이 모델들은 각 이온을 모형화하여 막 전위를 계산하는 HH 모델과 다르게, 스파이크 신호의 입력 패턴, 시간, 막 전위의 변화 등을 간단한 수식으로 나타냄으로써 연산량을 줄였다. 연산량이 적다는 특징으로 인해, HH 모델 보다 대규모 신경망을 시뮬레이션하는 과정에서 빈번하게 사용된다.

3. 기존 방식의 문제점 제기

LIF 모델 [7] 과 Izhikevich 모델 [3] 은 대규모 시뮬레이션 과정에서 HH 모델 [8] 에 비해 빈번하게 사용되지만, 신경세포의 제한된 특징만 설명할 수 있다는 단점이 있는데, 이를 보완하기 위해 LIF 모델 [7] 은 강조하려는 신경세포의 특징에 따라 다양한 변이 모델을 가지고 있다. 신경세포 모델의 다양성은 신경세포 모델의 특징이 전체적인 신경망의 연산 과정에서 어떤 방식으로 영향을 미치는지 확인하는 데 큰 도움을 준다. 현존하는 NEST [4], pyNN [5] 과 같은 SNN 시뮬레이션 구조는 다양한 신경세포 모델을 지원하기 위해 각 모델에 해당하는 서로 다른 코드를 제공한다. 그러나 이와 같은 프로그래밍 방법은 제한된 개수의 신경세포 모델만 지원하는 경우에는 효과적일 수 있지만, 지원하는 모델의 개수가 증가할수록 코드의 양이 선형적으로 증가하고 코드의 관리를 어렵게한다.

4. 해결책 및 실험 방식

3절에서 다룬 문제점은 주로 [9][10] 과 같이, SNN 시뮬레이션을 위한 하드웨어를 고려할 때 다뤄진다. 두 연구는 3가지 과정을 통해 다양한 신경세포 모델을 하드웨어 기반으로 지원하였다. 우선, 두 연구에서는 지원하려는 신경세포 모델을 제한된 몇 가지의 행동 양식으로 분류한다. 그 후, 분류된 행동 양식 각각에 해당하는 하드웨어를 설계한다. 마지막으로, 설계된 행동 양식을 조합해서 시뮬레이션하고자 하는 신경세포 모델을 구현한다.

본 연구에서는 위 연구들의 방법론이 소프트웨어 설계에 적용될 경우, 각 행동 양식에 대한 코드를 여러 신경세포 모델에서 재사용할 수 있으리라 예측하였다. 이를 검증하기 위해, 본 연구에서는 대표적인 소프트웨어 시뮬레이터 중 하나인 NEST에서 지원하는 LIF 모델 기반 신경세포 모델들의 코드를 분석하고, 각 모델에 존재하는 행동 양식을 구분했다. 또한, 분석 내용을 기반으로 적은 행동 양식에 대한 코드로 NEST에 존재하는 LIF 모델이 지원 가능하다는 것을 보여준다.

5. NEST 코드 분석 결과

NEST와 같은 소프트웨어 시뮬레이터는 복잡한 뉴런 모델의 행동 양식을 쉽게 표현하기 위해서 시간을 여러 시간 단위로 나누고 각 시간 단위마다 뉴런의 행동 방식을 정의한다. 위와 같은 방식을 clock-driven simulation이라고 하는데, 본 연구에서는 해당 simulation 방식에 초점을 맞춰서 프로그래밍 방법에 집중하였다. 또한, LIF 모델에서 사용되는 행동 양식을 분류하기 위해 12개의 LIF 모델을 분석하였다.

이를 통해 해당 12개의 모델은, 5개의 범주로 분류될 수 있는 13개의 행동 양식을 이용하여 표현될 수 있다는 것을 확인할 수 있었다. 본 절에서는 13개의 행동 양식을 5개의 범주를 통해서 순차적으로 분류하고, 각 LIF 모델에서 나타나는 행동 양식을 표 1에 정리한다. 또한, 분류 기준과 행동 양식을 5.1 ~ 5.5 절에서 순차적으로 설명한다(설명에서 나오는 모든 식은 가장 간단한 형태의 수식을 나타낸다.). 마지막으로, 각 행동 양식이 NEST의 코드에서 어떠한 방식으로 나타나는지 설명한다.

5.1. 막 전위 감소(Leakage)

신경세포의 막 전위는 입력 스파이크가 없으면 시간이 지남에 따라 감소한다. 이러한 과정은 신경세포 모델에 따라 다른 방식으로 표현되는데, NEST에서 분석한 LIF 모델의 경우 모든 신경세포 모델이 동일한 방식으로 막 전위 값을 감소시키는 것을 확인하였다. 다음은 해당 모델에서 공유하는 막 전위 감소 식이다. 아래 식에서 C는 축전량, g는 전도도, v는 막 전위 값, El은 외부 전압의 값을 나타낸다. NEST 시뮬레이터는 막 전위 감소를 나타내기 위해서 각 시간간격마다 막 전위에 1보다 작은 상숫값을 곱해준다 [11].

$$C\frac{dv}{dt} = -g(v - E_L)$$

5.2. 스파이크 축적(Accumulation)

신경세포에 들어오는 스파이크가 막 전위에 미치는 영향은 크게 전류 형태의 축적(current-based), 전도도 형태의 축적(conductance-based) 2가지로 나뉜다. 전류 형태의 축적 방식은 입력으로 들어오는 스파이크가 막 전위로 흘러 들어가는 전류의 크기를 직접 변화시킴으로써 막 전위의 크기를 변화시킨다. 반면, 전도도 형태의 축적 방식은 막 전위의 전도도 크기를 변화시키는 방식으로, 간접적으로 막 전위의 크기를 증가시킨다. 각각에 의한 막 전위 값의 변화는 아래 식을 통해 표현된다. 아래 식에서 I와 g_e는 각각 입력으로 전달되는 스파이크에 따라 변화하는 전류와 전도도를 나타낸다 [12][13].

$$current\ based: C\frac{dv}{dt} = I$$

$$conductance\ based: C\frac{dv}{dt} = g_e(v - E_e)$$

NEST 코드를 통해 분석한 12가지의 대표적인 LIF 모델은 둘 중 한 가지의 방식으로 스파이크 값을 축적하는 것을 확인할 수 있었다. 또한, 모델의 종류에 따라 I와 g_e의 값이

다른 방식으로 변화하는 것을 확인하였는데, I값은 스파이크에 의해 총 3가지: delta, exp, alpha 방식으로 값을 변화시키는 것을 확인할 수 있었고, g_e는 exp, alpha 2가지 방식으로 값을 변화시키는 것을 확인할 수 있었다. 즉, LIF 모델에서 입력으로 들어가는 스파이크에 의해 막 전위 값이 변화하는 방식은 총 5가지 행동 양식으로 구분될 수 있는 것을 확인하였다. Clock-driven 시뮬레이터에서는 전류 형태의 축적을 통해서 매 시간간격 입력으로 들어오는 전류의 값을 계산하고, 이를 막 전위의 값에 직접적으로 축적을 시킨다. 전도도 형태의 스파이크 축적은 전도도의 크기와 막 외부와 내부 사이의 전압 차이를 곱해서 매 시간간격 막 전위의 증가량을 계산한다.

5.3. 스파이크 발생(Spike Generation)

신경세포는 해당 세포의 막 전위가 특정 값을 넘어가게 되면 스파이크 신호를 발생시킨다. NEST에 존재하는 LIF 모델은 3가지 형태로 스파이크를 발생시킨다. 가장 기본적인 방식에서는 신경세포의 막 전위가 특정 값을 넘기면 스파이크를 발생시킨다. 두 번째 방식은 2개의 문턱 전압을 설정하고, 첫 번째 문턱 전압을 넘기면 막 전위의 크기를 기하급수적으로 증가시키는데, 이렇게 증가한 전위 값이 두 번째 문턱 전압을 초과하면 스파이크를 발생시킨다. 마지막 방식은, 이전 두 방식과 다르게, 현재 전위의 값을 기준으로 스파이크가 발생하는 확률을 계산한다. 본 연구에서는 각각을 absolute, exponential, probabilistic spike generation 방식이라 부른다.

$$absolute: if\ v > v_{th}\ \ generate\ spike$$

$$exponential: \frac{dv}{dt} = \Delta_T e^{\frac{v-v_{th}}{\Delta_T}}, if\ v > v_{th2}\ generate\ spike$$

$$probabilistic: rate = ReLu(c_1 v + c_2 \exp(c_3 v))$$

위 3가지 식은 각 스파이크 발생 방법을 수식적으로 나타낸 것이다. v_{th}, v_{th2}는 두 개의 문턱 전압을 나타내며, Δ_T, c_1, c_2, c_3는 상숫값이다. 또한, probabilistic 방식은 계산하여 나온 rate의 값에 비례한 확률로 스파이크를 발생시킨다.

5.4. 피드백 신호(Feedback)

신경세포에서는 연속적으로 스파이크가 발생하거나 들어오는 경우, 이후에 스파이크가 발생하는 것을 저해하는 피드백이 존재한다. LIF 모델은 Dynamic Threshold와 Adaptive Current, 두 가지 방식을 통해 피드백을 신경세포에 전달한다. Dynamic Threshold 방식은 5.3. 절에서 언급한 문턱 전압의 값을 변화시키는 방식이다. 만약 어떤 신경세포에서 스파이크가 연속적으로 발생하면, 해당 신경세포의 문턱 전압이 증가하고, 스파이크가 발생할 확률이 감소한다. 반대로, 스파이크가 오랜 기간 발생하지 않는다면, 문턱 전압의 크기는 감소하고, 스파이크가 발생할 확률은 올라간다. Adaptive Current 방식은 입력으로 스파이크가 전달될 때, 전달된 스파이크 개수에 비례하는 세기의 전류가 신경세포 밖으로 나가도록 발생시키는 방식이다. 이 두 가지 피드백 신호는 모두 SNN 내에서 스파이크가 균일한 정도로 발생할 수 있게 해준다. 위 두가지 방식은 event-driven 시뮬레이터에서 비슷한 방식으로 적용된다. 두 방식 모두, 해당 뉴런에서 스파이크가 발생하면 문턱 전압과 전류가 특정 값으로 변하게 되고 이 값은 스파이크가 발생하지 않는다면 막 전위 감소와 마찬가지로 1보다 작은 값이 곱해져 지수적으로 감소한다.

5.5. 휴지 기간(Refractory)

신경세포는 스파이크가 발생한 직후 일정 기간은 입력으로 스파이크를 받더라도 막 전위의 값이 변하지 않는다. 이러한 기간을 휴지 기간이라 부르는데, 이 역시 LIF 모델의 종류에 따라 다른 방식으로 처리한다. Absolute 방식은 스파이크가

Category	Behavior	Representative LIF model in NEST						
		iaf_psc (delta/exp/alpha)	iaf_cond (exp/alpha)	iaf_tum 2000	aeif_psc (exp/alpha)	aeif_cond (exp/alpha)	pp_psc_delta	gif_psc_exp
Leakage	Exponential Leak	O	O	O	O	O	O	O
Accumulation	Current based (delta)	O (delta)					O	
	Current based (exp)	O (exp)		O	O (exp)			O
	Current based (alpha)	O (alpha)			O (alpha)			
	Conductance based (exp)		O (exp)			O (exp)		
	Conductance based (alpha)		O (alpha)			O (alpha)		
Spike Generation	Absolute	O	O	O	O	O		
	Exponential				O	O		
	Probabilistic						O	O
Feedback	Dynamic Threshold						O	O
	Feedback Current				O	O	O	O
Refractory	Absolute	O	O	O	O	O	O	O
	Total			O				

표 1. NEST의 대표적 LIF 모델의 행동 양식 분석

발생한 직후, 정해진 시간 동안 막 전위의 크기가 변하지 않게 만든다. Total 방식은 Absolute 방식을 조금 더 발전시킨 방식이다. 두 개의 고정된 시간간격을 잡고, 첫 번째 시간간격이 지나기 전까지 막 전위의 값을 고정시킨다. 첫 번째 시간간격이 지난 이후에는 막 전위의 값은 변화하지만, 5.3. 절의 문턱 전압을 넘더라도 두 번째 시간간격이 지나기 전까지는 스파이크를 발생시키지 않는다. Clock-driven 시뮬레이터는 각 뉴런이 스파이크를 발생기킨 시간을 저장하고 이 시간을 기준으로 휴지 기간에 의한 영향을 적용시킨다.

표 1은 본 절의 분석 결과를 기반으로, NEST 시뮬레이터에 존재하는 모든 LIF 모델을 표현한 것이다. 이를 통해 하나의 행동 양식이 다양한 LIF 모델에 의해 공유되는 것을 확인할 수 있었다. 특히, Leakage, Absolute Refractory의 행동 양식은 모든 LIF 모델에 의해서 공유되는데, 이는 특정 행동 양식을 모형화 하는 코드가 매우 다양한 신경세포 모델에서 재사용 될 수 있는 것을 의미한다. 또한, 행동 양식 단위로 코드를 관리할 경우, 기존의 방법보다 효율적으로 다양한 모델에 대해서 코드를 유지하고 보수할 수 있음을 보여준다.

6. 결론 및 향후 연구 방향

본 연구는, SNN 시뮬레이션을 위한 신경세포를 소프트웨어로 모형화 할 때, 프로그래밍 코드를 유지, 보수 및 재사용하는 관점에서 효율적인 프로그래밍 방법론을 제시한다. 이를 위해, 범용성이 있는 SNN 시뮬레이터 중 하나인 NEST를 이용하여, 기존에 구현되어있는 신경망 모델 코드를 분석하고, 각 코드를 행동 양식 단위로 구분 짓는다. 또한, 위 과정을 통해 구분 지은 행동 양식이 다양한 신경세포 모델에서 공통으로 나타나는 것을 확인함으로써, 행동 양식 단위로 프로그래밍을 할 경우, 기존의 신경세포 모델 단위 프로그래밍보다 코드의 재사용이 많아져 효율적일 것이라 제안한다. 본 연구는, 추후 LIF 모델 이외에도 범용적으로 사용되는 Hodgkin-Huxley 모델 [8] 과 Izhikevich 모델 [3] 까지 포함할 수 있도록 연구를 확장할 예정이다.

Acknowledgement

본 연구는 IDEC의 EDA Tool을 지원받아 수행함. 또한 서울대학교 자동화시스템연구소와 반도체공동연구소의 지원을 받음.

참고문헌

[1] Diehl, Peter U., and Matthew Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity." Frontiers in computational neuroscience 9 (2015): 99.

[2] Clopath, Claudia, et al. "Connectivity reflects coding: a model of voltage-based STDP with homeostasis." Nature neuroscience 13.3 (2010): 344.

[3] Izhikevich, Eugene M. "Simple model of spiking neurons." IEEE Transactions on neural networks 14.6 (2003): 1569-1572.

[4] Gewaltig, Marc-Oliver, and Markus Diesmann. "Nest (neural simulation tool)." Scholarpedia 2.4 (2007): 1430.

[5] Davison, Andrew P., et al. "PyNN: a common interface for neuronal network simulators." Frontiers in neuroinformatics 2 (2009): 11.

[6] Hines, Michael L., and Nicholas T. Carnevale. "NEURON: a tool for neuroscientists." The neuroscientist 7.2 (2001): 123-135.

[7] Stein, Richard B. "A theoretical analysis of neuronal variability." Biophysical Journal 5.2 (1965): 173-194.

[8] Hodgkin, Alan L., and Andrew F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve." The Journal of physiology 117.4 (1952): 500-544.

[9] Lee, Dayeol, et al. "Flexon: a flexible digital neuron for efficient spiking neural network simulations." Proceedings of the 45th Annual International Symposium on Computer Architecture. IEEE Press, 2018.

[10] Frenkel, Charlotte, Jean-Didier Legat, and David Bol. "A compact phenomenological digital neuron implementing the 20 Izhikevich behaviors." Biomedical Circuits and Systems Conference (BioCAS), 2017 IEEE. IEEE, 2017.

[11] Dayan, Peter, and L. F. Abbott. "Theoretical neuroscience: computational and mathematical modeling of neural systems." Journal of Cognitive Neuroscience 15.1 (2003): 154-155.

[12] Rotter, Stefan, and Markus Diesmann. "Exact digital simulation of time-invariant linear systems with applications to neuronal modeling." Biological cybernetics 81.5-6 (1999): 381-402.

[13] Meffin, Hamish, Anthony N. Burkitt, and David B. Grayden. "An analytical model for the 'large, fluctuating synaptic conductance state' typical of neocortical neurons in vivo." Journal of computational neuroscience 16.2 (2004): 159-175.

대사증후군 예방을 위한 EMC 프로그램 개발

이수경

계명대학교

Soo1005s@gmail.com

Development of e-Motivate4Change (EMC) Program for Prevention of Metabolic Syndrome

Soo-Kyoung Lee

Keimyung University

요 약

The purpose of this study is to develop e-Motivate4Change (EMC) program to prevention of metabolic syndrome. The EMC program was developed using a software development life cycle method: analysis, design, implementation, and evaluation. The EMC program has been run on the web and mobile application. The EMC program enable to access and use much of the information you need to prevent the metabolic syndrome using health big data.

Key words: Big Data, Metabolic Syndrome, Mobile Application, Prevention, Wearable Device

1. 서 론

최근 우리나라는 생활양식의 변화에 따른 각종 만성질환이 증가되어 사람들의 만성질환 관리에 대한 관심이 높아지고 있다. 특히 대사증후군(metabolic syndrome, MetS)은 고혈압, 고중성지방혈증, 고혈당, 복부비만 등의 만성질환으로 발전할 수 있는 최대 위험요인으로 유병률이 증가하고 있는 추세이다[1]. 대사증후군의 경우 식습관과 운동, 흡연이나 음주, 스트레스 등 적극적인 생활습관 개선으로 위험요인을 감소시키고 심혈관 질환 발병을 예방할 수 있다[2].

정보 및 스마트 산업기술이 급격히 발전하면서 건강 애플리케이션(application, App)과 같은 모바일 기술을 활용하여 질병을 예방하고 증진하는 모바일 헬스(m-health)가 그 영역을 확장하고 있으며, 이에 대한 수요 또한 증가하고 있다[3].

앱스토어와 구글플레이에 설치된 헬스 앱의 숫자는 2013년도에 66,713개에서 2017년도에는 318,572개로 약 5배로 증가되었다[4]. 현재 건강관리에 도움을 줄 수 있는 다양한 건강 앱이 많이 출시되어 있으나 콘텐츠가 불만족스럽거나 사용이 불편하여 지속적으로 사용되지 않는 경우가 많다[5]. 대상자들이 지속적인 건강행위를 실천할 수 있도록 니즈(needs)를 정확하게 파악하여 이에 맞는 프로그램을 제공할 필요가 있다.

웨어러블 기기(wearable device, WD)는 언제 어디에서나 간편하게 신체활동을 모니터링 할 수 있을 뿐만 아니라 연동된 앱으로 맞춤형 피드백을 제공해준다. 실제로 국외 연구에서 효과를 검증한 결과, 웨어러블 기기가 생활 습관 개선에 긍정적인 영향을 미치는 것으로 나타났다[6]. 오늘날에는 Fitbit, 샤오미, 삼성 등 다양한 기업에서 웨어러블 기기를 출시하고 있으며, 그 중 헬스케어나 피트니스와 같은 건강관리를 목적으로 하는 기기가 시장에서 특히 빠르게 성장하고 있다[7]. 하지만 아직까지 국내에서는 대사증후군의 조기예방효과가 높은 성인초기 대상자에게 웨어러블 기기를 활용한 통합적인 대사증후군 예방 프로그램을 개발한 연구는 미비한 실정이다.

본 연구는 성인초기 대상자를 대상으로 대사증후군 예방 프로그램에 웨어러블 기기와 앱을 적극적으로 활용하고 재미와 동기부여 요소를 추가하여 기존의 대사증후군 예방 및 관리 프로그램의 한계점을 극복하고 개선하는데 기여하고자 한다. 이에 본 연구에서는 대학생을 대상으로 대사증후군 예방을 위한 e-Motivate4Change (EMC) 프로그램을 개발하고자 한다.

2. EMC 프로그램 개발 및 구현

EMC 프로그램은 맞춤형 상호작용 프로그램으로 대상자들이 자신의 정보를 입력하면 그에 따른 맞춤형 정보와 피드백을 제공하고 대상자들의 흥미 및 자기효능감을 상승시키고, 빅데이터 및 게임화 등의

기능을 접목시킨 성인초기 대상자들을 위한 대사증후군 예방 및 중재 프로그램이다.

본 연구에서는 성인초기 대학생의 대사증후군 인지 및 실천 증진을 위한 EMC 프로그램을 분석, 설계, 개발, 운영, 평가의 단계로 개발하였다. 분석단계에서는 관련 문헌고찰, 전문가 자문 등의 자료조사를 통해 EMC 구성항목을 결정하였으며, 설계 단계에서는 시스템 인터페이스와 스토리보드를 작성하여 EMC 설계를 하였다. 이를 바탕으로 관리자를 위한 반응형 웹(Responsive Web)과 사용자용 EMC 앱(App)을 개발하였다. 성인초기 대학생의 대사증후군 예방을 위한 EMC 프로그램의 구성은 그림 1과 같다.

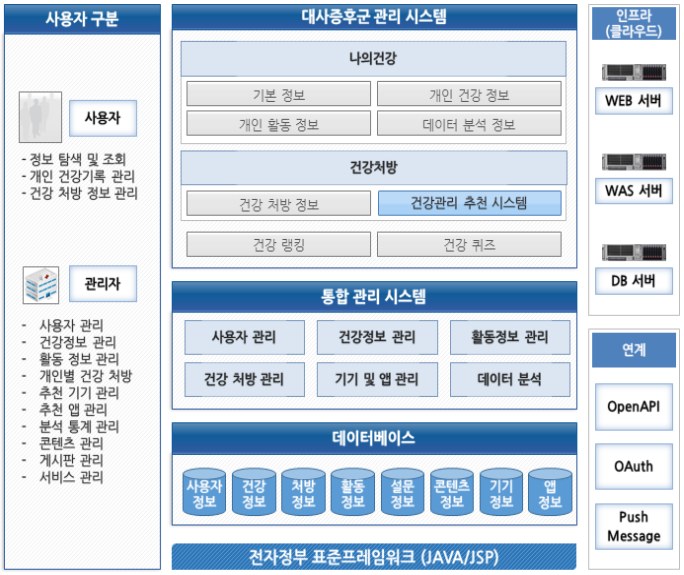


그림 1. System Architecture of EMC

본 연구의 성인초기 대학생의 대사증후군 인지 및 실천 증진을 위한 EMC 프로그램은 개인의 건강 행위 향상을 위한 중재로 구성하였다. 우선 개발된 앱의 초기화면에는 대사증후군 질환에 대한 간략한 소개를 볼 수 있다. 그리고 회원 가입 및 로그인 페이지에서는 대사증후군 상태를 진단할 수 있다. 현재 자신의 대사증후군 위험도는 어느 정도이며 대상자의 체형이미지도 한 눈에 볼 수 있도록 구성하였다. 대사증후군 상태 진단과 동시에 선호하는 건강관리 방법(앱, 웨어러블, 식이요법, 운동방법)에 대해서도 선택할 수 있도록 구성하였다.

대사증후군 관리시스템을 살펴보면 건강매니저를 통하여 건강관리 피드백을 제공하고 커뮤니티에서는 SNS와 질문 그리고 재미의 요소를 추가한 게임과 퀴즈로 구성되어있다. 그리고 본 연구 프로그램의 활성화와 연구 참여를 높이기 위해 실생활 알람 서비스를 추가하였고, 대상자의 건강관리 성과를 모니터하기 위해 대사증후군 위험도를 재평가하여 변화된 모습을 아바타로 볼 수 있도록 구성하였다. 마지막으로 본 추천 시스템 사용에 대한 평가와 만족도 차원으로 사용자 평가기능을 추가하였다.

EMC 시스템의 개체 관계도(entity relationship diagram, ERD)는 그림 2와 같다.

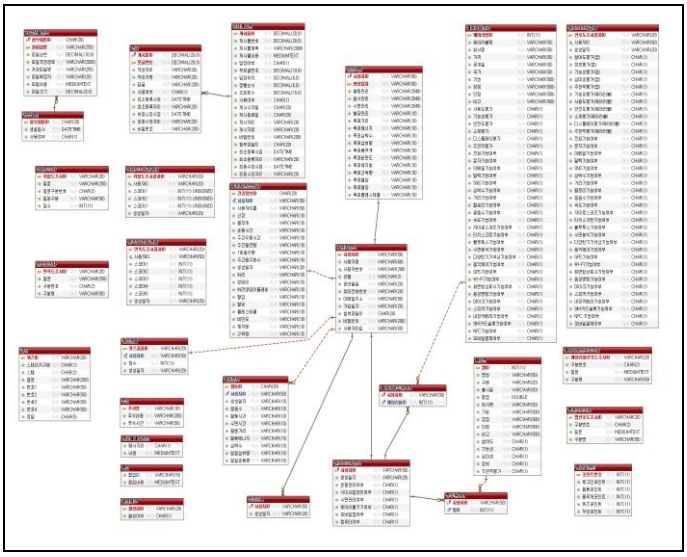


그림 2. Entity Relationship Diagram of EMC

성인초기 대상 대사증후군 인지 및 실천 증진을 위한 EMC 프로그램의 주요 사용자 화면은 그림 3과 같다.

(a) EMC 로그인 (b) 마이페이지 (c) 위험도 분석

(d) 선호도 조사 (e) 웨어러블 추천 (f) 건강처방전

그림 3. User Interface of EMC

EMC 프로그램의 구체적인 사용 시나리오는 다음과 같다. 처음에 사용자는 EMC 앱을 다운로드하여 본인의

스마트폰에 설치한 후 카카오톡(소셜 서비스 계정)을 이용하여 간편로그인을 한다. EMC 소개화면에서 EMC 목적과 사용방법, 대사증후군의 정의, 종류, 진단법, 위험요인, 관련 사이트와 동영상을 볼 수 있다. 나의 건강정보에 들어가 간단한 개인정보와 몸무게를 기입하고 대사증후군 위험도 분석을 실시한다. 대사증후군 위험도(생활습관, 식습관, 질병관련)가 점수로 보여지고, 나의 캐릭터가 표시된다.

사용자는 건강관리 선호도 조사를 통해 웨어러블 기기와 모바일 건강 앱을 추천 받는다. 대상자 개개인의 선호하는 건강관리 방법이 결정되면 대상자에게 적합한 웨어러블 기기와 건강 앱이 추천되고 이때 건강매니저들의 대상자별 맞춤형 건강처방전(운동관리, 식이요법관리, 수면관리 등)이 함께 제공된다. 사용자가 선택한 웨어러블 기기를 사용하여 본인의 활동데이터(심박수, 활동에너지, 걸음 수, 일일섭취량, 활동거리, 활동시간, 수면시간)를 정해진 요일에 직접 입력하거나 건강매니저에게 전송한다. 이때 입력된 데이터는 건강매니저에 의해 관리된다. 정기적으로 건강매니저의 관리 하에 신체계측(키, 몸무게, 체성분)과 혈압측정 및 혈액검사(혈당, 콜레스테롤)를 측정한다. 측정된 검사결과는 건강매니저들이 대상자 개인 건강정보에 직접 입력 후 관리하고 새롭게 입력된 데이터는 새로운 처방전을 작성한다. 사용자는 새로 제시된 건강처방을 제공받는다. 건강 퀴즈를 풀거나 게시판 활동 등을 하면 건강포인트가 적립되어 건강랭킹에서 나의 순위를 확인할 수 있다.

3. 결 론

본 연구는 생활습관이 이미 고착화된 중장년층이 아닌 예방차원의 접근으로 IT에 익숙한 성인초기 대상자들이 쉽고 재미있게 접근할 수 있고, 선호도를 고려함으로써 실제적으로 실효성 있는 방법들을 유헬스 기반의 실천성과 접근성이 높은 중재프로그램을 개발하여 적용하였다는 점에서 의의가 있으며, 웨어러블 기기를 적극적으로 활용하고 앱에 동기부여 요소를 포함시켜 기존의 대사증후군 중재 프로그램의 한계점을 극복하고 개선하는데 기여하였다.

본 연구에서 개발된 프로그램은 성인초기 대학생의 대사증후군 예방 효과뿐만 아니라 유사 생활 습관 관련 만성질환에 대한 지역사회 예방 및 관리 사업에서의 비용 효율적인 전략 구성에 있어 기초 자료로 활용될 것이다. 자기관리를 통해 대사증후군 발생이 감소하기 때문에 치료, 검사 등의 요구가 줄어들게 되므로 개인적으로나 의료기관 측면에서도 의료비용을 절감을 할 수 있다. 대사증후군 발생과 입원 감소를 유도하여 성공적 건강전환을 통한 삶의 질 향상에 기여할 것이다.

Acknowledgement

본 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구 (No. NRF-2015R1C1A1A01055753)이며, 또한 2018년도 정부 (교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2018R1D1A3B07050652).

참고문헌

[1] Korea Centers for Disease Control and Prevention. Community Health Survey. Available from: http://kosis.kr/common/meta_onedepth.jsp?vwcd=MT_OTITLE&listid=117_A01. 2016.

[2] K. H. Nam, J. Y. Kang, Y. J. Lee, S. H. Sung, Y. K. Jang, J. Y. Lee, I. K. Park, and T. I. Choi, Journal of Nutrition and Health. Comparison of health outcomes according to intervention compliance in male Korean workers with metabolic syndrome. Vol.46, No.2, pp.156-165. 2013.

[3] Y. J. Kim, Journal of the Korea Contents Association. Exploratory Study on Acceptance Intention of Mobile Devices and Applications for Healthcare Services. Vol.12, No.9, pp.369-379. 2012.

[4] Aitken, M., C. Brian, and N. The Growing Value of Digital Health. IQVIA Institute for Human Data Science, 2017.

[5] E. Seo, Study on smart mobile healthcare UX design for persistent motivation. Unpublished master's thesis. Seoul University, Seoul. 2015.

[6] J. B. Wang, L. A. Cadmus-Bertram, L. Natarajan, M. M. White, H. Madanat, J. F. Nichols, G. X. Ayala and J. P. Pierce, Telemedicine and e-Health. Wearable sensor/device (Fitbit One) and SMS text-messaging prompts to increase physical activity in overweight and obese adults: A randomized controlled trial. Vol.21, No.10, pp.782-792. 2015

[7] National information society agency, Wearable device based creative economy revitalization strategy. No.6. 2014.

모놀리식 애플리케이션의 컨테이너 기반 마이크로서비스 전환을 위한 스크립트 생성 방법

김대호°*, 윤동규*, 박준석**, 염근혁§교신저자
부산대학교 전기전자컴퓨터공학과*§, 부산대학교 물류혁신네트워킹연구소**
{suitable, lodestar692, pjs50, yeom}@pusan.ac.kr

Script Generation Method to convert Monolithic Applications into Container-Based Microservice

Daeho Kim°*, Donggyu Yun*, Joonseok Park**, Keunhyuk Yeom§corresponding author
Department of Electrical and Computer Engineering Pusan National University*§,
Research Institute of Logistics Innovation and Networking**

요 약

본 논문에서는 기존 모놀리식 애플리케이션의 설계 자료 분석하여 컨테이너 오케스트레이션(Container Orchestration) 환경에 배포 및 관리할 수 있는 템플릿 스크립트를 자동으로 생성하는 방법을 제시하였다. 기존 모놀리식 애플리케이션을 마이크로서비스 단위의 컨테이너 기반으로 전환하려 할 때 배포 및 관리에 필요한 구성 요소들을 수동으로 매핑하여 정의해야하는 문제점을 가지고 있다. 본 논문에서는 UML의 수집 요소를 정의하고 이를 실제 컨테이너 오케스트레이션 환경인 Kubernetes의 템플릿 스크립트를 생성하는 사례 연구와 이를 실행하여 마이크로서비스 배포 및 관리가 가능함을 검증하였다.

1. 서론

컨테이너 오케스트레이션(Container Orchestration)이란 다수의 컨테이너의 생성, 배포 등 전반적인 컨테이너 라이프사이클을 일괄적으로 관리할 수 있는 기술이다.

최근 컨테이너 오케스트레이션 기술을 접목시켜 기존 모놀리식 애플리케이션을 마이크로서비스 단위로 재구성한 뒤 컨테이너 기반 환경으로 배포를 시도하고 있다[1]. 그러나 초기 배포 및 관리에 필요한 구성 요소들을 기존 모놀리식 설계 자료에서 수동적으로 분석 및 매핑해야 하는 문제점이 발생한다[2].

본 논문에서는 모놀리식 애플리케이션에서 재구성된 마이크로서비스를 컨테이너 오케스트레이션 환경으로 배포할 수 있도록 템플릿 스크립트를 자동으로 생성하는 방법을 제시하였다. 또한 사례 연구로 Kubernetes[3] 환경에서 템플릿 스크립트를 생성하는 과정을 보였고 이를 실행하여 마이크로서비스 생성 및 배포 관리가 가능함을 검증하였다.

2. 마이크로서비스 배포를 지원하는 템플릿 스크립트 생성 방법

컨테이너 오케스트레이션 환경인 Kubernetes 환경에서 관리 및 배포에 필요한 주요 템플릿 스크립트 명세 모델은 그림 1와 같다. Common Info는 모든 종류의 템플릿 스크립트에서 기술해야하는 요소를 의미한다. Pod는 Kubernetes에서 컨테이너 정보가 담긴 요소를 의미한다. ReplicaSet은 다수의 Pod를 제어하고 관리할 수 있는 정보가 담긴 요소를 의미한다. Service는 생성된 Pod 서비스가 외부 클라이언트와 통신하기 위한 네트워크 정보가 담긴 요소를 의미한다.

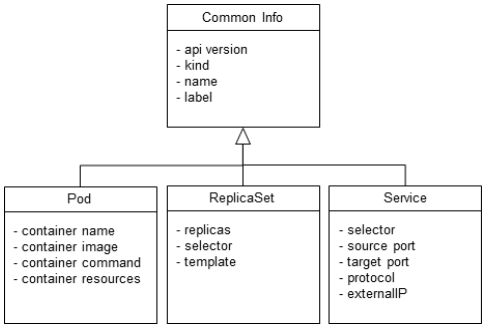


그림 1 Kubernetes 주요 템플릿 스크립트 모델

모델에서 정의한 요소와 모놀리식 설계 자료에서 정의된 수집 요소를 연결하는 매핑 테이블을 표 1과 같이 정의하였다. Pod는 마이크로서비스 이름과 노드 정보에서 수집된 요소들을 매핑하여 스크립트를 생성한다. 여기서 마이크로서비스는 모놀리식의 클래스 설계 자료를 기반으로 도출[4]되었음을 전제로 한다. ReplicaSet은 클래스 설계 이름과 구성된

"이 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2017R1D1A1B03030243)"

마이크로서비스 수 그리고 컨테이너 템플릿 정보를 생성하기 위한 노드 정보를 매핑하여 생성한다. 마지막으로 Service는 배치 설계 자료에서 수집된 연결 정보를 매핑하여 스크립트가 생성한다.

표 1 템플릿 스크립트 생성 매핑 테이블

설계 종류	설계 요소	수집 요소	Pod	ReplicaSet	Service
클래스 설계 자료	마이크로서비스 정보	마이크로 서비스 이름	name		
		마이크로 서비스 수	-	replicas	
	클래스 정보	클래스 설계 이름	label	name, selector, label	name, selector, label
배치 설계 자료	노드 정보	노드 이름	container name	template (container name)	
		노드 OS	container image	template (container image)	
		노드 리소스	container resource	template (container resource)	
		노드 Application	container command	template (container command)	
	연결 정보	외부 포트			source port
		내부 포트			target port
		연결 프로토콜			protocol
		외부 IP			external IP

3. 사례 연구

표 1에서 정의한 매핑 테이블을 기반으로 실제 설계 자료인 대학정보시스템 UML 설계 자료를 적용하여 템플릿 스크립트를 생성하는 과정을 보였다. 그림 2는 대학 정보 시스템의 UML 배치 설계 자료에서 매핑테이블을 기반으로 추출하여 종류별 템플릿 스크립트를 생성하는 과정이다.

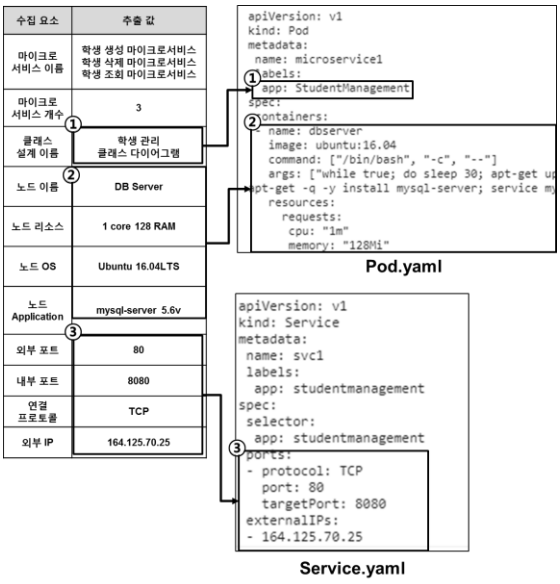


그림 2 종류별 템플릿 스크립트 생성

①번으로 표시된 영역은 클래스 다이어그램 정보를 추출하여 Pod, ReplicaSet, Service 템플릿 스크립트의 Label을 생성하는 과정이다. ②번으로 표시된 영역은 배치 다이어그램의 노드 정보를 추출하여 Pod, ReplicaSet 템플릿 스크립트의 container 및 template을 생성하는 과정이다. ③번으로 표시된 영역은 연결 정보를 추출하여 service 템플릿 스크립트의 모든 요소를 생성하는 과정이다. 그림 3은 그림 2에서 생성된 Pod 스크립트를 Kubernetes 환경(1개의 Master , 3개의 Node 각각 2core 2RAM 할당 및 구성)에서 실행하였을 때 나타나는 결과이며 이들이 올바르게 동작함을 확인할 수 있다.

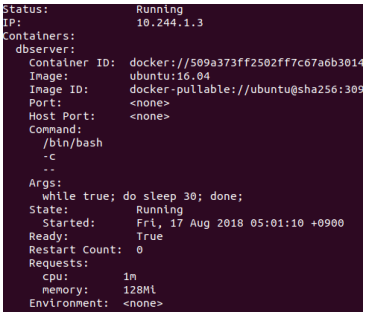


그림 3 Pod 템플릿 스크립트 실행 결과

5. 결론 및 향후 연구

본 논문에서는 모놀리식 애플리케이션을 마이크로서비스로 전환하여 컨테이너 오케스트레이션 환경으로 배포 할 수 있도록 템플릿 스크립트 자동으로 생성하는 방법을 제시하였다. 본 논문에서 제시된 방법은 모놀리식 애플리케이션을 컨테이너 기반 마이크로서비스 환경으로 전환할 수 있는 기반 모델이 될 것으로 기대된다. 향후 연구로는 컨테이너 관리 모니터링 도구에 대한 연구를 진행할 것이다.

참고문헌

[1] 5 best practices for container orchestration in IT production, https://techbeacon.com/5-best-practices-container-orchestration-it-production

[2] E. Casalicchi, “Autonomic orchestration of containers: Problem definition and research challenges.”, 10th EAI International Conference on Performance Evaluation Methodologies and Tools, Taormina, Italy, October. 2016.

[3] Kubernetes : Cloud Providers, https://kubernetes.io/docs/concepts/cluster-administration/cloud-providers/

[4] 김대호, 박준석, 염근혁, “모놀리식 애플리케이션의 UML 설계 자료에 기반한 마이크로서비스 구성 방법”, 한국차세대컴퓨팅학회 논문지, 제 14 권 5 호, pp 7-18, 2018. 10.

GAS 절감이 가능한 스마트 컨트랙트 코드 패턴 분석 방법

정다은○ 박수용
서강대학교 컴퓨터공학과
judaen44@sogang.ac.kr, sypark@sogang.ac.kr

Method for analyzing smart contracts code patterns being capable of saving GAS

Daeun Jeong○ Sooyong Park
Sogang Univ.

요 약

본 연구에서는 GAS 절감이 가능한 스마트 컨트랙트 코드 패턴 분석 방법을 다루고 있다. 이더리움의 스마트 컨트랙트 코드에 GAS를 많이 소모하는 특정 패턴들이 포함된 경우에는 GAS를 낭비하게 되는 문제가 생길 수 있다. 이를 위해 기존에 제시되었던 패턴 외에 새로운 두 가지 패턴과 그 패턴을 검출하는 방법을 제시한다. 이더리움 블록체인 네트워크의 스마트 컨트랙트 분석 방법은, 스마트 컨트랙트의 코드를 파싱하는 단계, 파싱된 스마트 컨트랙트 코드로부터 GAS 소모가 많은 사전 설정된 패턴을 검출하는 단계, 사전 설정된 패턴이 검출된 결과를 출력하는 단계로 이루어져 있다. 패턴 검출 단계에서 검출할 패턴은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴이다. 스마트 컨트랙트 코드에서 GAS 소모가 많은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴을 검출하여 GAS 소모가 작은 패턴으로 개선하도록 함으로써, GAS 소모를 줄일 수 있다.

1. 서 론

블록체인(Blockchain) 기술은 P2P 네트워크를 이루는 노드들이 네트워크에 전송하는 트랜잭션들을 블록에 저장하고, 각 노드들에 전술한 블록들을 체인 형태로 저장하는 기술이다. 2013년 러시아 출신 캐나다 개발자인 비탈릭 부테린이 "차세대 스마트 컨트랙트 & 분산 응용 애플리케이션 플랫폼"이라는 제목으로 이더리움 백서를 발간했고[1], 차세대 스마트 컨트랙트는 이더리움의 가장 중요한 특징 중 하나이다. 스마트 컨트랙트(Smart Contract)는 코드에 적힌 계약 조건이 만족되면 즉시 계약이 성사되도록 구성되어 있으며, 스크립트 언어인 솔리디티(Solidity)로 작성 가능하다. 이와 같이 솔리디티로 작성된 스마트 컨트랙트 코드가 컴파일되어 바이트 코드가 생성되며, 이 바이트 코드는 이더리움 클라이언트를 통해 블록에 기록된다. 블록체인 네트워크에 참여하는 노드들이 해당 스마트 컨트랙트를 이더리움 가상 머신(Ethereum Virtual Machine; 'EVM')으로 실행시킬 수 있게 된다. 블록에 기록된 스마트 컨트랙트를 EVM으로 실행할 때 GAS를 지불해야 된다. GAS는 이더리움에서 스마트 컨트랙트를 실행시키는 대가로서 지불되는 네트워크 수수료이자 연산량의 대가이기도 하다. 대부분의 블록체인은 데이터를 저장하고 동일한 연산을 중복되게 실시하므로 사용할 수 있는 자원에 한계가 있다. 이더리움은 계약의 실행을 위한 연산력이 한정된 자원이므로 이를 사용하는 대가로서 가스(GAS)를 지불해야 한다. 그런데, 스마트 컨트랙트 코드에 GAS를 많이 소모하는 특정 패턴들이 포함된 경우에는 GAS를 낭비하게 되는 문제가 발생하게 된다. 이러한 최적화되지 않은 스마트 컨트랙트 코드는 코드의 생성자 및 사용자가 GAS 비용을 필요 이상으로 부담하게 한다.

2. 기존 연구

기존에 제시된 GAS를 낭비하게 되는 여러 패턴들은 크게 *1. Useless Code Related Patterns, 2. Loop Related Patterns* 두 가지로 나뉜다[2].

*Useless Code Related Patterns*에서는 기존의 일반 프로그래밍 랭귀지에서 구동 시간이나 저장 공간의 낭비 패턴으로도 알려져 있는 *Dead Code*와 *Opaque Predicate*가 있다. *Dead Code*의 경우 조건문 a 안에 또 다른 조건문 b가 있을 때, a의 조건을 만족하면 무조건 b의 조건문을 만족할 수 없는 경우, b 조건문 안의 실행될 코드는 bytecode로 바뀌어서 EVM에 올라가게 되고 실행되지 않더라도 업로드 되는 비용을 사용자가 지불해야 한다. *Opaque Predicate*의 경우는 *Dead Code*와 반대로 조건문 a의 조건을 만족하면 무조건 b의 조건문도 만족할 경우, 조건문 b는 쓸모가 없음에도 불구하고 b 조건문 b도 bytecode로 바뀌어 EVM에 올라가므로 마찬가지로 비용을 낭비하게 한다.

*Loop Related Patterns*에는 1)*Expensive operations in a loop*, 2)*Constant Outcome of a loop*, 3)*Loop fusion*, 4)*Related computations in a loop*, 그리고 5)*Comparison with unilateral outcome in a loop*가 있다. 각각을 간단히 설명하자면, *1)Expensive operations in a loop*는 표 1의

Ethereum 백서에 정의된 Operation들 중 유난히 비싼 operation을 굳이 반복문 안에 배치하여 gas를 심히 낭비할 수 있는 패턴이다. *2)Constant Outcome of a loop*는 반복문 안의 실행 코드가 굳이 반복문을 돌리지 않아도 같은 결과값을 가지기 때문에 gas가 낭비되는 패턴이다. *3)Loop fusion*은 두 반복문의 실행 내용이 겹치지 않고 두 반복문의 반복 횟수가 똑같을 때, 한 반복문 안에 실행 내용을 합치지 않아 gas를 낭비시킬 수 있는 패턴이다. *4)Related computation in a loop*는 *1)Expensive operations in a loop*와 *2)Constant Outcome of a loop* 두 상황이 한번에 일어나는 경우로, 반복문의 결과가 항상 일정하고 반복문 안의 연산이 비싼 연산일 경우이다. 마지막 *5)Comparison with unilateral outcome in a loop*는 반복문 안에 있는 조건문의 항상 TRUE[or FALSE]일 경우여서 불필요한 gas를 낭비하는 패턴이다.

표 1 Gas Cost Of Different Operations

Operation	Gas	Description
ADD/SUB	3	Arithmetic operation
MUL/DIV	5	
ADDMOD/MULMOD	8	
AND/OR/XOR	3	Bitwise logic operation
LT/GT/SLT/EQ	3	Comparison operation
POP	2	Stack operation
PUSH/DUP/SWAP	3	
MLOAD/MSTORE	3	Memory operation
JUMP	8	Unconditional jump
JUMPI	10	Conditional jump
SLOAD	200	Storage operation
SSTORE	5,000/ 20,000	
BALANCE	400	Get balance of an account
CREATE	32,000	Create a new account using CREATE
CALL	25,000	Create a new account using CALL

3. 연구 내용

이러한 기존 연구에서 발견되지 않은 분리형 증감문과 AND 연산자 다중 조건문 패턴을 정의하고 가스 절감 방법을 소개하려 한다. 전술한 문제점을 해결하기 위한 본 연구의 목적은 스마트 컨트랙트 코드를 분석하여 분리형 증감문과 ADN 연산자 다중 조건문과 같은 패턴이 포함되어있는지 여부를 판단하고 보고함으로써, GAS를 절약시킬 수 있도록 하는 스마트 컨트랙트 분석 장치 및 방법을 제공하는 것이다.

전술한 기술적 과제를 달성하기 위한 본 연구의 스마트 컨트랙트 분석 장치는, 그림 1에서 볼 수 있는 것처럼 스마트 컨트랙트의 코드를 파싱하는 코드 파싱부; 상기 파싱된 스마트 컨트랙트 코드로부터 GAS 소모가 많은 사전 설정된 패턴들을 검출하는 패턴 검출부; 및 상기 패턴 검출부에 의한 검출 결과를 출력하는 패턴 보고서 출력부;를 구비하고, 상기 사전 설정된 패턴은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴을 포함한다. 스마트 컨트랙트 분석 장치에 있어서, 상기 패턴 검출부는, 그림 2에 자세히 나와있는 것처럼 상기 파싱된 스마트 컨트랙트 코드에 'for 구문' 이 있는지 여부를 판단하고, 'for 구문'에 증감문이 있는지 여부를 판단하고, 증감되는 루프(loop) 변수가 'for 구문'의 Body에 사용되었는지 여부를 판단하여, 상기 스마트 컨트랙트 코드에서 분리형 증감문 패턴을 검출하는 것이 바람직하다.

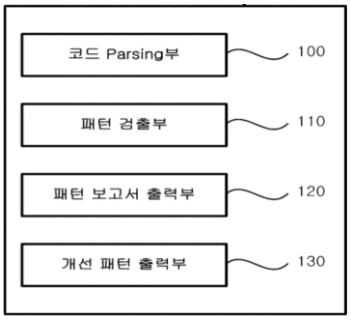


그림 1 패턴 검출과정 도식화 (simple.ver).

패턴 검출부는, 상기 파싱된 스마트 컨트랙트 코드에 조건문이 있는지 여부를 판단하고, 상기 조건문이 다중 조건문인지 여부를 판단하고, 다중 조건문이 AND로만 이루어져있는지 여부를 판단하여, 상기 스마트 컨트랙트 코드에서 AND 연산자 다중조건문 패턴을 검출한다.

본 연구의 GAS 절감이 가능한 스마트 컨트랙트 분석 방법은, 블록체인 네트워크의 스마트 컨트랙트 분석 장치에 의한 스마트 컨트랙트 분석 방법으로서, (a) 스마트 컨트랙트의 코드를 파싱하는 단계; (b) 상기 파싱된 스마트 컨트랙트 코드로부터 GAS 소모가 많은 사전 설정된 패턴을 검출하는 단계; (c) 상기 (b)단계에서 사전 설정된 패턴이 검출된 결과를 출력하는 단계; 를구비하고, 상기 사전 설정된 패턴은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴이다.

GAS 절감이 가능한 스마트 컨트랙트 분석 방법에 있어서, 상기 (b) 단계는, 상기 파싱된 스마트 컨트랙트 코드에 'for 구문'이 있는지 여부를 판단하고, 'for 구문'에 증감문이 있는지 여부를 판단하고, 증감되는 루프(loop) 변수가 'for 구문'의 Body에 사용되었는지 여부를 판단하여, 상기스마트 컨트랙트 코드에서 분리형 증감문 패턴을 검출하는 것이 바람직하다.

GAS 절감이 가능한 스마트 컨트랙트 분석방법에 있어서, 상기 (c) 단계는, 상기 파싱된 스마트 컨트랙트 코드에 조건문이 있는지 여부를 판단하고, 상기 조건문이 다중 조건문인지 여부를 판단하고, 다중조건문이 AND로만 이루어져있는지 여부를 판단하여, 상기 스마트 컨트랙트 코드에서 AND 연산자 다중 조건문 패턴을 검출한다.

GAS 절감이 가능한 스마트 컨트랙트 분석방법은, (e) GAS 소모가 많은 사전 설정된 패턴이 검출된 경우, 검출된 패턴과 동일한 기능을 수행하면서도 GAS 소모가 작은 개선 패턴을 출력하는 단계;를 더 구비하는 것이 바람직하다.

본 연구의 스마트 컨트랙트 분석 장치는, 스마트 컨트랙트

코드에서 GAS 소모가 많은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴을 검출하고 이를 보고하여 사용자로 하여금 GAS 소모가 작은 패턴으로 개선하도록 함으로써, GAS 소모를 절약할 수 있도록 한다.

첨부된 그림을 참조하여 본 연구의 스마트 컨트랙트 분석 장치 및 방법에 대하여 구체적으로 설명한다.

그림 1은 본 연구의 스마트 컨트랙트 분석 장치를 개략적으로 도시한 블록도이며, 그림 2는 본 연구의 스마트 컨트랙트 분석 장치에 있어서, 스마트 컨트랙트 분석 과정을 전체적으로 도시한 흐름도이다.

그림 1 및 그림 2를 참조하면, 본 연구의 스마트 컨트랙트 분석 장치(1)는 코드 파싱부(100), 패턴 검출부(110), 패턴 보고서 출력부(120) 및 개선 패턴 출력부(130)을 구비한다.

상기 코드 파싱부(100)는, 스마트 컨트랙트의 코드를 파싱(parsing)하여 제공한다.

상기 패턴 검출부(110)는, 상기 파싱된 스마트 컨트랙트 코드로부터 GAS 소모가 많은 사전 설정된 패턴들을 검출한다. 상기 사전 설정된 패턴은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴인 것이 바람직하다.

그림 3은 본 연구의 스마트 컨트랙트 분석 장치에 있어서, GAS 소모가 많은 제1 패턴인 분리형 증감문 패턴에 대한 예제 코드이며, 그림 5는 GAS 소모가 많은 제2 패턴인 AND 연산자 다중 조건문 패턴에 대한 예제코드이다.

상기 패턴 보고서 출력부(120)는, 상기 패턴 검출부에 의한 검출 결과를 출력하여 제공한다. 상기 개선 패턴 출력부(130)는, 상기 패턴 검출부에 의해 GAS 소모가 많은 사전 설정된 패턴이 검출된 경우, 검출된 패턴과 동일한 기능을 수행하면서도 GAS 소모가 작은 개선 패턴을 출력한다.

그림 2를 참조하여 상기 패턴 검출부의 동작에 대하여 보다 구체적으로 설명한다.

상기 패턴 검출부는, 스마트 컨트랙트 코드에서 제1 패턴인 분리형 증감문 패턴을 검출하기 위하여, 상기 파싱된 스마트 컨트랙트 코드에 'for 구문'이 있는지 여부를 판단하고, 'for 구문'에 증감문이 있는지 여부를 판단하고, 증감되는 루프(loop) 변수가 'for 구문'의 Body에 사용되었는지 여부를 판단한다.

이러한 과정을 거쳐, 코드 내에 증감문이 있는 'for 구문'이 있고, 증감되는 루프 변수가 'for 구문'의 Body에 사용된 경우, 상기 스마트 컨트랙트 코드에서 제1 패턴인 분리형 증감문 패턴을 검출한다.

또한, 상기 패턴 검출부는, 스마트 컨트랙트 코드에서 제2 패턴인 AND 연산자 다중 조건문 패턴을 검출하기 위하여, 상기 파싱된 스마트 컨트랙트 코드에 조건문이 있는지 여부를 판단하고, 상기 조건문이 다중 조건문인지 여부를 판단하고, 다중 조건문이 AND로만 이루어져있는지 여부를 판단한다.

이러한 과정을 거쳐, 코드 내에 다중 조건문이 있고 다중 조건문이 AND 만으로 이루어진 경우, 상기 스마트 컨트랙트 코드에서 제2 패턴인 AND 연산자 다중 조건문 패턴을 검출하게 된다.

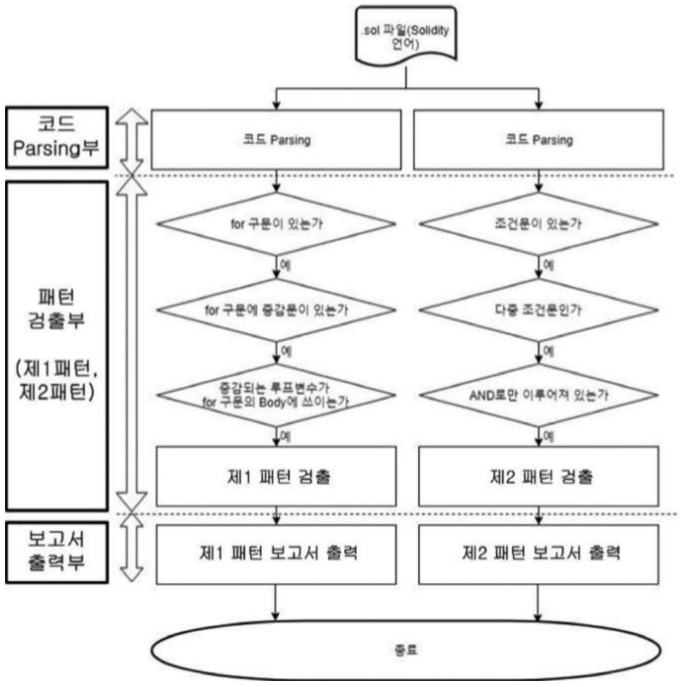


그림 2 패턴 검출과정 도식화 (concrete.ver)

상기 개선 패턴 출력부는, 상기 패턴 검출부에 의해 GAS 소모가 많은 사전 설정된 패턴들이 검출된 경우, 검출된 패턴과 동일한 기능을 수행하면서도 GAS 소모가 작은 개선 패턴을 출력하게 된다.

```
function p1 (uint x) public
{
    uint a=0;
    ...
        for(uint i=0;i<x;i++)
        {
        ...
        a += i;
        }
    ...
}
```

그림 1 Pattern 1 코드화

```
function p1_report (uint x) public
{
    uint a=0;
    ...
        for(uint i=0;i<x;)
        {
        ...
        a += i++;
        }
    ...
}
```

그림 2 Pattern 1 개선 코드화

3.1 패턴 1 실시 예

그림 4는 본 연구의 스마트 컨트랙트 분석 장치에 있어서, 그림 3의 제1 패턴에 대해 제시되는 개선 코드이다. 그림 4를 참조하면, 상기 제1 패턴에 대한 개선 패턴은, 제1 패턴에 있는 증감문을 Body에 있는 루프 변수에 one-line 형식으로 병합하도록 함으로써, GAS를 절약할 수 있게 된다.

3.2 패턴 2 실시 예

그림 6은 본 연구의 스마트 컨트랙트 분석 장치에 있어서, 그림 5의 제2 패턴에 대해 제시되는 개선 코드이다. 그림 6을 참조하면, 상기 제2 패턴에 대한 개선 패턴은, 제2 패턴의 다중 조건문을 (AND 연산자의 수)+1 개의 조건문으로 분리하여 Depth를 만듦으로써, GAS를 절약할 수 있게 된다.

이상에서 본 연구에 대하여 그 바람직한 실시 예를 중심으로 설명하였으나, 이는 단지 예시일 뿐 본 연구를 한정하는 것이 아니며, 본 연구가 속하는 분야의 통상의 지식을 가진 자라면 본 연구의 본질적인 특성을 벗어나지 않는 범위에서 이상에 예시되지 않은 여러 가지의 변형과 응용이 가능함을 알 수 있을 것이다. 그리고, 이러한 변형과 응용에 관계된 차이점들은 본 연구의 범위에 포함되는 것으로 해석되어야 할 것이다.

```
function p2 (uint x) public
{
    if( x>1 && x<10 )
    {
        ...
    }
}
```

그림 3 Pattern 2 코드화

```
function p2_report (uint x) public
{
    if( x>1 )
    {
        if( x<10 )
        {
        ...
        }
    }
}
```

그림 4 Pattern 2 개선 코드화

4. 실험 결과

제시한 패턴의 GAS 절감 효과를 확인하기 위해 우분투환경에서 Truffle과 Node.js, Reat,js, 그리고 Ganache를 설치하여 실험용 스마트 컨트랙트 코드인 TestGasConsumption.sol을 생성했다. 그 후 스마트 컨트랙트 코드의 메소드 별로 소모되는 GAS를 분석해주는 'eth-gas-reporter' 플러그인을 이용하기 위해 npm run test 명령어를 입력하였다. 각 메소드 별 가스 소모 내역이 표로 출력되었고 이를 비교한 결과는 아래와 같다.

표 2 Gas Cost 비교(그림5 소비코드 VS 개선코드)

A 조건문	B 조건문	소비코드(GAS)	개선코드(GAS)
TRUE	TRUE	100	95
TRUE	FALSE	86	81
FALSE	-	75	55

```
if(A){
    if(B){
        ...
    }
}
```

```
if(A && B){
    ...
}
```

그림 5 Pattern 2 GAS 비교 코드(좌:소비코드, 우:개선코드)

표 2의 실험 결과에 따르면 그림 5의 좌 소비코드와 우 개선코드가 조건문의 TRUE, FALSE 여부와 상관없이 조건문의 분리보다 다중조건문이 가스를 더 절감한다는 것을 알 수 있다.

5. 분석 결과

본 연구의 스마트 컨트랙트 분석 장치는, 스마트 컨트랙트 코드에서 GAS 소모가 많은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴을 검출하고 이를 보고하게 된다. 한편, GAS 소모가 많은 패턴이 검출되었음을 보고함과 동시에, GAS 소모가 작은 개선된 패턴을 알려줌으로써, 사용자로 하여금 GAS 소모가 작은 패턴으로 개선할 수 있도록 하여, 스마트 컨트랙트의 실행에 따른 GAS소모를 절약할 수 있도록 한다.

6. 결 론

본 연구에서는 GAS 절감이 가능한 스마트 컨트랙트 분석 방법을 다루었다. 이더리움 블록체인 네트워크의 스마트 컨트랙트 분석 방법은, 스마트 컨트랙트의 코드를 파싱하는 단계, 파싱된 스마트 컨트랙트 코드로부터 GAS 소모가 많은 사전 설정된 패턴을 검출하는 단계, 사전 설정된 패턴이 검출된 결과를 출력하는 단계를 구비하고, 패턴 검출 단계에서 검출할 사전 설정된 패턴은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴이다. 스마트 컨트랙트 코드에서 GAS 소모가 많은 분리형 증감문 패턴 또는 AND 연산자 다중 조건문 패턴을 검출하여 GAS 소모가 작은 패턴으로 개선하도록 함으로써, GAS 소모를 절약할 수 있다.

참고 문헌

[1]BUTERIN, V. A next generation smart contract & decentralized application platform. https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf/, retrieved Feb. 2015, 2013.

[2]T. Chen, X. Li, X. Lou, X. Zhang, Under-optimized smart contracts devour your money. IEEE 24th International Conference on Software Analysis, Evolution and Reengineerin, SANER, 2017, pp.442-446.

솔리디티 스마트 컨트랙트의 스크립트 기반 시나리오 테스트

문현아 ○ 박수용
서강대학교 컴퓨터공학과
{hamoon,sypark}@sogang.ac.kr

최광훈
전남대학교 전자컴퓨터공학부
kwanghoon.choi@jnu.ac.kr

A Script-based Scenario Test for Smart Contracts in Solidity

HyeonAh Moon○ Sooyong Park
Dept. of Computer Science
Sogang University

KwangHoon Choi
Dept. of Elect. and Computer Science
Chonnam National University

요 약

본 연구에서는 솔리디티 언어로 작성한 스마트 컨트랙트의 안정성 테스트를 위해 프로그램의 동작 시나리오를 쉽게 기술하고 직접 실행 가능하도록 설계한 스크립트 언어 이더스크립트(EtherScript)를 제안한다. 예제 시나리오를 통해 분산 환경에서 동작하는 스마트 컨트랙트의 특성을 고려하고, 이더의 흐름을 나타내는 다양한 시나리오를 구현하기에 적절한 언어임을 확인하였다.

1. 서 론

블록체인은 2008년 비트코인이 공개된 이후, 중개기관에 의존하지 않고 거래 당사자들간의 직접 거래를 가능하게 하는 기술로 많은 관심을 받고 있다. 특히, 이더리움 블록체인에서 처음 상용화된 스마트 컨트랙트는 블록체인 산업의 새로운 가능성을 보여주는 특성이다. 다양한 응용프로그램을 블록체인 플랫폼에서 사용할 수 있게 하는데, 예를 들면 가장 활발히 사용되는 ICO(Initial Coin Offering)을 위한 스마트 컨트랙트나 고양이를 사고 팔 수 있는 게임을 제작할 수도 있다. 이처럼 스마트 컨트랙트는 다양한 비즈니스 프로세스에 대응할 수 있지만 기존의 응용 프로그램들과는 다르게 탈중앙화 분산 환경에서 작동하는 특성으로 인해 프로그래머가 의도치 않은 오류가 발생할 가능성이 있다. 게다가 화폐와 밀접한 특성 또한 가지고 있어서 DAO공격[1]이나 Parity Wallet 사건[2][3]에서와 같이 프로그램의 작은 오류가 큰 자산의 손실을 부르기도 한다. 스마트 컨트랙트의 또다른 특성은 한번 배포된 후에는 수정이 불가능하다는 점이다. 이런 모든 특징들을 고려할 때 스마트 컨트랙트 개발에서의 테스트의 중요성은 더욱 증대된다.

스마트 컨트랙트를 개발한 후 메인 넷에 배포해 실제 테스트한다는 것은 리스크가 크고, 테스트 넷 테스트는 불필요한 지연 시간 등이 필요하므로 이전에 테스트 환경을 구축해 실행하는 것이 좋다. 사설망을 생성해 테스트 하는 경우는 사설망 생성부터 계정 생성 그리고 이후 계정의 비밀번호 관리까지 테스트와 관련없는 불필요한 작업에 시간을 많이 소모한다.

현재 개발에 많이 활용하는 트러플(Truffle) 프레임워크를 이용한 테스트는 자바스크립트와 솔리더티(Solidity) 모두 테스트 작성에 사용할 수 있다. 이 방식은 단위 테스트에는 유용하나 시나리오를 테스트 하려면 자바스크립트 문장 작성이 복잡하고 어렵다.

본 논문에서는 화폐의 가치를 지닌 이더의 흐름을 잘 기술하고 스마트 컨트랙트 프로그램의 동작을 표현하는 시나리오를 작성하기 쉽게 하는 이더스크립트 언어를 설계하고 구현하였다. 이 언어를 실제 스마트 컨트랙트의 시나리오 테스트에 적용하여 유용함을 확인하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를 살펴보고, 3장에서는 스마트 컨트랙트 시나리오 테스트를 위한 스크립트 언어인 이더스크립트의 설계 동기와 실제 적용 결과를 평가하고 이더스크립트 해석기의 구조를 제시한다. 마지막으로 4장에서는 결론과 향후 연구 방향에 대해 논의한다.

2. 관련 연구

현재 스마트 컨트랙트의 안정성을 위해 트러플(Truffle)[4], 슬리터(Slither: 사용자 개입없이 몇 초 안에 실제 취약점을 발견 할 수 있는 오픈 소스 정적 분석 프레임 워크)[5], 에키드나(Echidna: 추상적 상태 머신 모델링 및 최소한의 자동 테스트 케이스

생성과 같은 강력한 기능을 갖추고 있는 스마트 퍼저)[6], 오인트(Oyente: 심볼릭 실행을 이용한 취약점 분석도구)[7], 스마트첵(SmartCheck: 알려진 취약성 및 나쁜 습관을 온라인에서 자동으로 검사하고 코드에서 해당 부분을 강조 표시하고 문제에 대한 자세한 설명을 제공)[8], 미스릴(Mythril: concolic 분석, 오염 분석 및 제어 흐름 검사를 사용하여 다양한 보안 취약점을 탐지)[9] 등의 여러 테스트 및 디버깅 도구와 서드파티 감사 서비스를 이용할 수 있다. 그러나 이 도구들은 시나리오 테스트와 이더 흐름 분석에는 적절하지 않다.

특히, CLI를 이용하거나 리믹스(Remix)에서 시나리오 테스트 하는 경우는 1) 시나리오에 필요한 만큼 계정 생성, 시나리오에 따라 2) 어떤 계정에서 스마트 컨트랙트를 배포하고 3) 어떤 계정으로 스마트 컨트랙트 내의 함수를 호출할 것인지 등을 매번 바꾸어 주는 것은 매우 번거롭고, 다시 테스트 해야 하는 경우 같은 작업을 반복해야 한다.

3. 이더스크립트: 스마트 컨트랙트 시나리오 테스트를 위한 스크립트 언어

이 절에서는, 이더리움 기반 스마트 컨트랙트 프로그램의 동작 시나리오를 쉽게 기술하고 직접 실행 가능하도록 설계한 스크립트 언어 이더스크립트(EtherScript)를 제안한다.

3.1 이더스크립트 언어를 설계한 동기

스마트 컨트랙트 프로그램은 여러 계정 또는 다른 스마트 컨트랙트와 자유롭게 이더(ether)를 주고받거나 함수를 호출 하면서 동작하도록 구성되어 있어 시나리오 테스트를 준비하는 과정이 더 복잡하다. 주어진 시나리오에 따라 이 프로그램을 테스트하려면 첫째, 시나리오에 관여하는 계정을 생성하고 적절한 이더를 보유하도록 초기화하고, 시나리오에서 연동할 다른 스마트 컨트랙트도 추가로 설치해야 한다. 둘째, 이렇게 준비된 계정들과 스마트 컨트랙트들이 정해진 시나리오에 따라 이더를 주고받고 함수를 호출하는 순서를 정하고 이 순서대로 실행한다.

솔리디티 기반 스마트컨트랙트 프로그램의 개발환경 리믹스에서 시나리오 테스트를 진행하려면 웹 기반 사용자 인터페이스를 통해 텍스트 창에 계정 주소, 함수 인자, 송금할 이더 등을 입력하고 버튼을 눌러 실행하는 과정을 반복해야 한다. 시나리오 기반 회귀 테스트를 위해 이러한 작업은 자동화되어야 한다.

대표적인 이더리움 클라이언트인 고이더리움(Geth)나 이더리움제이(Ethereumj)에서 계정 만들기, 스마트 컨트랙트 설치하기, 송금, 함수 호출을 범용 프로그래밍언어인 자바스크립트와 자바 프로그램을 실행하여 수행할 수 있다. 하지만 간단한 시나리오를 테스트하기 위해서 상당히 긴 프로그램을 작성해야한다. 게다가 솔리디티 프로그램을 작성하는 프로그래머가 자바스크립트나 자바 언어에 능숙 해야 하는 문제가 있다.

따라서 이러한 문제를 개선하고자 솔리디티 프로그램의 시나리오를 쉽게 기술하고 자동으로 반복 실행하기 적합하며 솔리디티 언어와 유사한 구문으로 작성할 수 있는 스크립트 언어를 제안한다.

3.2 스크립트 언어 및 적용 사례

솔리디티 프로그램에 대한 시나리오를 실행하도록 작성한 스크립트 프로그램 사례를 들어 이더스크립트 언어를 소개한다.

```
// simple_storage.sol
contract SimpleStorage {
 uint public storedData;
 function set(uint x) public {   storedData = x; }
 Function get() public view returns (uint)
   {    return storedData;     }

// simple_storage.es
Line1: account{balance:10ether} owner;
Line2: account{balance:50ether} user;
Line3: account{contract:“simple_storage.sol”,
          by:owner}  simplestorage{“SimpleStorage”};
Line4: simplestorage.set(123) {by:user};
Line5: uint x;
Line6: x = simplestorage.get() {by:user};
Line7: assert x==123;
```

그림 1. 솔리디티 프로그램과 이더스크립트 프로그램

[그림 1]의 simple_storage.sol은 컨트랙트 SimpleStorage를 선언한 솔리디티 프로그램이다. 이 컨트랙트는 uint 타입의 상태 변수 storedData와 이 변수 값을 설정하고 읽는 두 개의 함수 set과 get으로 구성되어 있다.

[그림 1]의 다음 예제 simple_storage.es는 본 논문에서 제안한 이더스크립트로 작성한 프로그램이다. 앞의 컨트랙트에서 선언한 두 함수의 동작을 테스트하기 위해 간단한 시나리오를 작성하였다. 첫 번째와 두 번째 줄에서 컨트랙트를 설치할 계정과 이 컨트랙트를 사용할 계정으로 owner와 user를 선언한다. 각 계정을 선언할 때 balance 속성으로 초기 보유할 이더를 지정한다. 세 번째 줄에서 [그림 1]의 솔리디티 프로그램 이름, 컨트랙트를 생성한 계정, 컨트랙트 이름을 지정하여 컨트랙트 계정 simplestorage를 만든다. 이 계정을 통해 컨트랙트에서 제공하는 함수를

호출할 수 있다. 네번째 줄에서 set함수로 123을 상태 변수에 설정하고, 여섯 번째 줄에서 get함수를 통해 다시 읽어온 다음, 일곱 번째 줄에서 assert문으로 확인한다.

본 연구에서 이더스크립트 해석기를 구현하였다. 스크립트 프로그램을 실행하면 먼저 이더리움 블록체인을 새로 만들고, 스크립트 프로그램에 선언된 계정을 생성한 다음 프로그램 문장들을 차례대로 실행하도록 해석기를 구성하였다.

이더스크립트 활용 사례로 이더리움의 대표적인 취약점으로 알려진 다오(DAO) 공격 시나리오와 블록체인과 스마트 컨트랙트를 활용하는 예탁 판매(Escrow) 시나리오를 작성하였다.

```
// dao.sol
contract SimpleDAO {
  mapping (address => uint) public credit;
  function donate(address to) payable {
    credit[to] += msg.value;
  }
  function queryCredit(address to) constant returns (uint)
  { return credit[to]; }
  function withdraw(uint amount) {
    if (credit[msg.sender] >= amount) {
      msg.sender.call.value(amount)();
      credit[msg.sender] -= amount;
}}}

contract Mallory {
  SimpleDAO public dao;
  address owner;
  function Mallory(SimpleDAO _dao) {
    dao = _dao; owner = msg.sender;
  }
  function() payable
    { dao.withdraw(dao.queryCredit(this));  }
  function getJackpot()
    { owner.send(this.balance);  }
}
```

그림 2. 다오 컨트랙트와 맬러리 컨트랙트

[그림 2]에서 컨트랙트 SimpleDAO는 여러 사용자가 기부한 재원을 모으는 프로그램이고, 컨트랙트 Mallory는 다오 취약점 공격에서 해커가 사용할 프로그램이다. 이 컨트랙트들과 다오 취약점에 대한 자세한 설명은 [10]을 참고한다.

[그림 3]은 이 다오 취약점을 공격하는 시나리오를 이더스크립트로 작성한 프로그램이다. 1번째 줄에서 5번째 줄까지 이 시나리오에 관여하는 계정을 만들고, 6번째 줄에서 다오 컨트랙트 SimpleDao의 계정 dao를 생성한다. 이 계정의 donate함수를 통해서 사용자 계정 user1, user2, user3에서 1이더씩을 이 dao 계정에 보낸다. 10번째 줄에서 해커는 컨트랙트 Mallory 계정 mallory를 만들고 11번째 줄에서 100피니를 먼저 다오 계정에 기부한 다음 12번째 줄에서 Mallory 컨트랙트의 폴백(fallback) 함수를 호출하여 다오의 재진입 취약점(reentrancy vulnerability)을 공격하여 다오 계정에 세가지 사용자 계정에서 기부한 3이더를 mallroy 계정으로 뺏고, 13번째 줄에서 이 돈을 해커 계정으로 옮겨온다. 14번째 줄에서 해커 계정의 금액을 assert문을 통해 출력하여 확인한다.

```
Line01: account{balance:50ether} owner;
Line02: account{balance:50ether} user1;
Line03: account{balance:50ether} user2;
Line04: account{balance:50ether} user3;
Line05: account{balance:50ether} hacker;
Line06: account{contract:"dao.sol", by:owner}
          dao("SimpleDAO");
Line07: dao.donate(user1) {by:user1, value:1ether};
Line08: dao.donate(user2) {by:user2, value:1ether};
Line09: dao.donate(user3) {by:user3, value:1ether};
Line10: account{contract:"dao.sol", by:hacker}
          mallory("Mallory", dao);
Line11: dao.donate(mallory) {by:hacker, value:100finney};
Line12: mallory.() {by:hacker};
Line13: mallory.getJackpot() {by:hacker};
Line14: assert hacker.balance;
```

그림 3. 다오 취약점 공격 시나리오를 구현한 스크립트

아래 GitHub 프로젝트에서 지금까지 소개한 두 가지 예제를 포함해서 예탁 판매와 경매 예제 (King of the Throne) 관련 솔리디티 프로그램과 이더스크립트 프로그램을 내려 받을 수 있다.

- https://github.com/kwanghoon/ethereumj/tree/working

3.3 평가

이 논문에서 제안한 이더스크립트를 제대로 평가하기에 아직 초기 개발 단계이지만 현재까지 연구를 수행한 경험을 바탕으로 정량적 평가와 정성적 평가를 수행하였다.

정량적 평가는 3가지 동일한 시나리오에 대해 이더스크립트 언어로 작성한 스크립트 프로그램과 이더리움제이 플랫폼 상에서 자바로 작성한 프로그램을 LOC(Line Of Count) 기준을 측정하여 시나리오 작성의 용이한 정도를 비교하였다. 이 자바 프로그램도 위 GitHub 사이트에서 찾아볼 수 있다.

표 1. LOC 비교

LOC	EtherScript	Java (EthereumJ)
Simple Storage	8	47
Dao	18	82
Escrow	13	62

동일한 시나리오를 이더스크립트 언어를 사용하면 3가지 사례를 기준으로 평균 20%에 해당하는 LOC 정도로 작성할 수 있었다. 자바 프로그램의 LOC를 셀 때 주석이나 공백뿐만 아니라 이더스크립트 프로그램과 직접적인 비교 대상이 아닌 줄도 모두 제외하였기 때문에 이 프로그램의 실제 LOC는 더 늘어난다. 이러한 점을 비교해보면 이더스크립트로 시나리오를 더 간단하게 작성할 수 있음을 확인하였다.

정성적 측면에서 이더스크립트는 두 가지 장점을 가지고 있다. 첫째, 이더리움제이 플랫폼에서 자바로 시나리오를 작성한다면 두 가지 언어를 알아야 하는 부담이 있다. 솔리디티 프로그래머가 테스트하고 싶은 시나리오를 직접 작성하도록 하기 위해서 이더스크립트는 솔리디티 언어와 유사한 구문을 사용하도록 설계하였다. 특히 화폐의 가치가 있는 이더를 주고받는 이더리움 도메인에 특정적인 새로운 스크립트 언어로 솔리디티 프로그램에서 가장 중요한 특징인 이더의 흐름을 시나리오로 구현하기 매우 간단하다. 즉, 솔리디티 사용자에게 익숙한 계정 생성, 함수 호출을 통한 계정간의 이더 전송, 각 계좌의 잔고 확인을 위한 이더스크립트 구문을 이용하여 쉽게 구현할 수 있다.

둘째, 리믹스와 같은 솔리디티 프로그래밍 환경에서 테스트하는 방법과 비교하면 이더스크립트 프로그램은 자동으로 반복 실행이 쉬운 장점을 가지고 있다. 따라서 회귀 테스트에 더 적합하다.

3.4 이더스크립트 해석기 구현

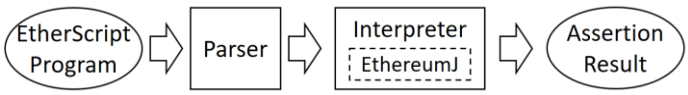


그림 4. 이더스크립트 해석기 구성

이더스크립트 해석기는 크게 파서와 해석 모듈로 구성되어 있다. 솔리디티 언어의 구문 정의를 최대한 참조하되 이더스크립트에만 있는 계정 선언 구문과 컨트랙트 계정 선언 구문을 추가하였다.

해석 모듈은, 파서 모듈에서 출력한 이더스크립트 추상구문트리를 따라 순서대로 한 문장씩 실행하는 구조이다. 스크립트 프로그램 실행 전에 이더리움 블록체인을 생성하고, 일반 계정과 컨트랙트 계정 생성, 송금이나 컨트랙트 함수 호출을 구현하기 위해서 이더리움제이에서 제공하는 라이브러리를 활용하였다.

4. 결론 및 향후 연구

본 논문에서 이더리움 솔리디티 프로그램의 동작에 관한 시나리오를 쉽게 표현할 수 있는 이더스크립트 언어를 설계하고 구현하였다. 이 언어를 사용하여 대표적인 스마트 컨트랙트 활용 시나리오를 작성한 경험에 대해 평가하였다. 화폐 가치가 있는 이더를 주고받는 이더리움 도메인을 고려하여 설계한 언어이기 때문에 이더 흐름에 대한 시나리오를 구현하는데 적절한 언어임을 확인하였다.

향후 연구로, 첫째, 더 많은 사례 연구를 통해 이더스크립트 언어의 활용도를 평가하고자 한다. 둘째, 솔리디티 프로그램과 시나리오에 대한 모델을 바탕으로 테스트 시나리오를 자동 생성하고 이 시나리오를 이더스크립트로 작성하여 테스트하는 방법에 대한 연구를 수행하고자 한다.

참고문헌

[1] Analysis of the DAO exploit. http://hackingdistributed.com/2016/06/18/analysis-of-thedao-exploit/. Last access, 2018.

[2] The Parity Wallet Hack Explained. https://blog.zeppelin.solutions/on-the-parity-wallet-multisighack- 405a8c12e8f7. Last access, 2018

[3] The Wallet Smart Contract Frozen by the Parity Bug. https://github.com/paritytech/parity/blob/4d08e7b0aec46443bf26547b17d10cb302672835/js/src/contracts/snippets/enhanced-wallet.sol. Last access, 2018.

[4] TRUFFLE SUITE. https://truffleframework.com/. Last access, 2018.

[5] Slither.https://blog.trailofbits.com/2018/10/19/slither-a-solidity-static-analysis-framework/. Last access, 2018.

[6] Echidna.https://blog.trailofbits.com/2018/03/09/echidna-a-smart-fuzzer-for-ethereum/. Last access, 2018.

[7] L. Luu, D.H. Chu, H. Olickel, et al. Making Smart Contracts Smarter. In Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security, pp:254-269, Vienna, Austria, 2016.

[8] SmartCheck. https://tool.smartdec.net/. Last access, 2018.

[9] Mythril. https://mythril.ai/. Last access, 2018.

[10] N. Atzei, M. Bartoletti, T. Cimoli. A Survey of Attacks on Ethereum Smart Contracts. In proceedings of the 6th International Conference on Principles of Security and Trust, pp: 164-186, Berlin, Heidelberg, 2017.

보안성을 고려한 경량 크래시 리포트 생성 기법

장동민 ○, 이수준, 장유원*, 정호현, 이은석
성균관대학교 전자전기컴퓨터공학과, 성균관대학교 소프트웨어플랫폼학과*
{jangdm, hoakw, jangyoowon, jeonghh89, leees}@skku.edu

Lightweight crash report generation technique considering security

Dongmin Jang○, Sujune Lee, Yoowon Jang*, Hohyeon Jeong, Eunseok Lee
Department of Electrical and Computer Engineering, Sungkyunkwan University
Department of Software Platform, Sungkyunkwan University*

요 약

소프트웨어는 배포 후 처리되지 않은 예외 혹은 결함으로 인해 크래시가 발생할 수 있다. 개발자가 이와 같은 오류를 추적하고 수정하는 디버깅 작업은 소프트웨어의 품질을 향상시키는데 매우 중요하다. 하지만 다양한 환경에서 배포된 소프트웨어인 경우 오류 추적 작업이 매우 어려운데, 그 이유는 결함 발생 시점의 실행 환경 정보를 무분별하게 개발자에게 전송하기 때문이다. 이러한 무분별한 정보로 인해 분석 난이도가 높아 오류 추적 비용이 많이 발생하고, 개인정보의 보안을 유지하기 어려운 문제가 있다. 본 논문에서는 이와 같은 문제를 해결하기 위해 오류 추적에 유용한 정보만 수집 및 가공하여 보안성이 높은 고품질 경량 크래시 리포트를 생성 기법을 제안한다. 제안 기법의 유효성을 확인하기 위해 오픈 소스 프로젝트에 적용하여 기존 연구와 동일 수준의 오류 추적 정보를 제공하면서 정보양을 72% 감소하였다.

1. 서론

소프트웨어가 배포된 이후 실행 환경에서 예외로 인한 크래시가 발생하면 운영체제에서 크래시 리포트를 생성하여 개발자에게 전송한다 [1,2]. 크래시 리포트를 수신한 개발자는 오류 추적 및 오류 수정 작업 후 모든 테스트를 통과하면 소프트웨어를 재배포하는데 이 모든 과정을 디버깅 작업이라고 한다. 디버깅에 대한 많은 연구가 진행되고 있으나 본 논문에서 다루는 크래시 리포트 생성 단계에 다음과 같은 한계점이 있다.

기존의 크래시 리포트 생성 단계는 오류 발생 시 운영체제에서 시스템 정보, 메모리 덤프, 크래시 덤프 등을 포함한 덤프 파일을 첨부하여 크래시 리포트를 생성한다 [3,4,5]. 이 중 덤프 파일을 이용하는 도구인 WinDbg [5]는 덤프 파일의 정보를 메모리 덤프 정보, 크래시 덤프, 콜 스택 정보, 어셈블리 등으로 추출하여 개발자에게 텍스트 정보로 제공한다.

이러한 크래시 리포트는 실행환경에 대한 정보를 개발자가 확인하는 장점이 있지만, 덤프 파일의 정보 중 오류 추적에 필수로 확인하지 않아도 될 데이터까지 포함되어 분석 비용이 높은 크래시 리포트를 받을 수 있다. 또한, 첨부되는 데이터 중에서 사용자가 사용하는 다른 모듈의 정보 및 개인 정보가 포함될 가능성이 있어 보안 측면에서 취약한 문제점이 존재한다.

본 논문에서는 위와 같은 한계들을 극복하기 위해 개인 정보를 제외한 정보를 크래시 리포트에 제공하여 보안성을 고려하고, 오류 추적에 도움이 되는 정보로만 구성되어 분석 비용이 낮은 경량 크래시 리포트 생성 기법을 제안한다.

2. 제안 방법

2.1 개요

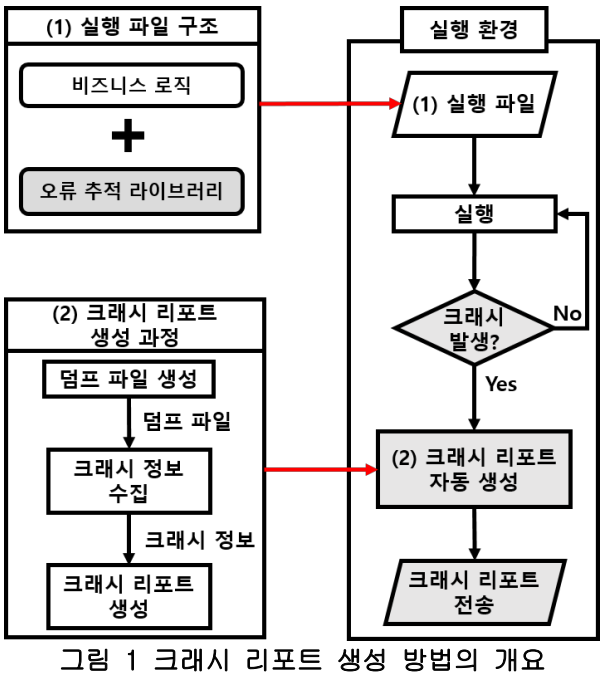


그림 1 크래시 리포트 생성 방법의 개요

그림 1은 제안하는 기법의 순서도로 본 논문에서 다루는 크래시 리포트 자동 생성에 대한 상세 과정을 표현한다.

개발자가 실행 파일을 배포하기 전에 라이브러리 형태로 만든 제안 기술을 개발 소프트웨어에 삽입하고 라이브러리 호출하는 코드 한 줄을 작성하고 배포한다. 실행 환경에서 배포된 소프트웨어를 실행하는 도중 크래시가 발생하면 소프트웨어에 포함된 라이브러리에서 크래시 리포트 자동 생성 단계를 수행한다. 해당 단계에서는 1) 덤프 파일 생성, 2) 오류 추적용 필수 정보 수집, 3) 크래시 리포트 생성하여 크래시 리포트 전송 작업을 수행한다.

오류 추적 수집 정보		
DUMP_CLASS	APPLICATION_VERIFIER_FLAGS	**FAILURE_MODULE_NAME**
DUMP_QUALIFIER	PRODUCT_TYPE	BUCKET_ID_MODULE_STR
FAULTING_IP	SUITE_MASK	**FAILURE_FUNCTION_NAME**
EXCEPTION_RECORD	DUMP_TYPE	BUCKET_ID_FUNCTION_STR
Parameter	ANALYSIS_SESSION_HOST	BUCKET_ID_OFFSET
FAULTING_THREAD	ANALYSIS_SESSION_TIME	BUCKET_ID_MODTIMEDATESTAMP
BUGCHECK_STR	ANALYSIS_VERSION	BUCKET_ID_MODCHECKSUM
DEFAULT_BUCKET_ID	THREAD_ATTRIBUTES	BUCKET_ID_MODVER_STR
PROCESS_NAME	**PROBLEM_CLASSES**	BUCKET_ID_PREFIX_STR
ERROR_CODE	PRIMARY_PROBLEM_CLASS	FAILURE_PROBLEM_CLAS
EXCEPTION_CODE	LAST_CONTROL_TRANSFER	FAILURE_SYMBOL_NAME
EXCEPTION_CODE_STR	**STACK_TEXT**	FAILURE_BUCKET_ID
EXCEPTION_PARAMETER	THREAD_SHA1_HASH_MOD_FUNC	TARGET_TIME
WATSON_BKT_PROCSTAMP	THREAD_SHA1_HASH_MOD_FUNC_OFFSET	OSBUILD
WATSON_BKT_MODULE	THREAD_SHA1_HASH_MOD	OSSERVICEPACK
WATSON_BKT_MODSTAMP	**FOLLOWUP_IP**	SERVICEPACK_NUMBER
WATSON_BKT_MODOFFSET	FAULT_INSTR_CODE	OS_REVISION
WATSON_BKT_MODVER	SYMBOL_STACK_INDEX	**OSPLATFORM_TYPE**
BUILD_VERSION_STRING	SYMBOL_NAME	**OSNAME**
MODLIST_WITH_TSCHKSUM_HASH	FOLLOWUP_NAME	**OSEDITION**
MODLIST_SHA1_HASH	**MODULE_NAME**	USER_LCID
NTGLOBALFLAG	IMAGE_NAME	OSBUILD_TIMESTAMP
PROCESS_BAM_CURRENT_THROTTLED	**DEBUG_FLR_IMAGE_TIMESTAMP**	BUILDDATESTAMP_STR
PROCESS_BAM_PREVIOUS_THROTTLED	STACK_COMMAND	BUILDLAB_STR
FAILURE_ID_HASH	BUCKET_ID	**BUILDOSVER_STR**
BUCKET_ID_IMAGE_STR	FAILURE_EXCEPTION_CODE	ANALYSIS_SESSION_ELAPSED_TIME
Followup	FAILURE_IMAGE_NAME	ANALYSIS_SOURCE
FAILURE_ID_HASH_STRING		

그림 2 기존 크래시 리포트의 수집 정보와 제안한 크래시 리포트의 수집 정보(음영) 비교

2.2 덤프 파일 내 크래시 정보 수집

그림 2는 기존 덤프 파일에서 얻을 수 있는 정보의 리스트와 리스트 중 본 논문에서 제안하는 크래시 리포트에 필요한 정보(음영)를 나타낸 것이다. 제안하는 크래시 리포트는 덤프 파일로부터 크게 4가지 종류의 정보를 수집한다.

첫 번째 정보는 시스템 정보로 개발자가 오류 재현을 하기 위해 필요한 실행 환경의 환경 정보를 확인할 수 있다. 수집되는 정보는 운영체제 종류, 운영체제 버전, 프로세서의 레벨, 프로세서 아키텍처 정보를 수집한다. 두 번째 정보는 모듈 정보로 크래시 상황이 발생한 모듈의 정보를 수집한다. 해당 정보들은 실행 파일이나 동적 라이브러리와 같은 모듈의 이름, 모듈이 실제 메모리에 로드된 물리 주소 및 크기 정보, 개발 환경에서 일치하는 모듈을 찾는데 사용하는 타임스탬프 정보를 포함한다. 세 번째 정보는 스레드 정보로 크래시 상황이 발생한 지점까지의 실행 이력을 담은 정보를 수집한다. 이 정보들은 예외 상황이 발생한 스레드 ID, 콜 스택 정보이다. 마지막 정보는 예외 정보로 실제 개발에서 어떤 종류의 예외 상황이 발생했는지 구체적으로 알 수 있는 오류 발생 스레드 ID, 오류가 발생한 메모리 주소, 오류 이름(예: Null pointer exception)을 포함한다.

따라서 위에서 언급한 4개의 오류 추적 카테고리 중 23가지 정보를 활용하여 오류의 원인을 분석한다. 이는 그림 2의 기존 크래시 리포트 수집 정보 중 약 28%만 사용하여 WinDbg [5]와 동일한 수준의 오류추적을 할 수 있다. 그러므로 그 외에 나머지 정보는 실제 오류의 원인을 분석하거나 디버깅하는데 필수로 확인해야 하는 정보가 아니다. 또한, 필터링 된 정보 중 개인 정보가 포함되어 있는 정보 및 오류와 관련 없는 모듈의 정보도 제외하여 제안한 크래시 리포트의 보안성도 향상한다.

3. 실험

3.1 실험 프로젝트 및 평가 방법

제안하는 기술의 유효성을 평가하기 위해 Microsoft Windows 환경에서 실험을 진행하고 기존 오류 추적 도구인 WinDbg [5]와 비교한다. 실험을 위해 Native Win32 API 기반의 C++ 오픈 소스 프로젝트 [6]에 Null pointer dereference 또는 Exception access violation과 같은 크래시를 자주 발생시키는 오류 [7]를 주입하여 크래시 발생 시 크래시 리포트 생성 및 분석 실험을 진행한다. 성공적인 오류 추적의 평가 기준은 다음과 같다.

- 발생한 오류의 종류 분석
- 오류가 발생한 프로젝트의 모듈 및 소스 파일 탐지
- 오류가 발생한 소스 코드 라인 위치 식별

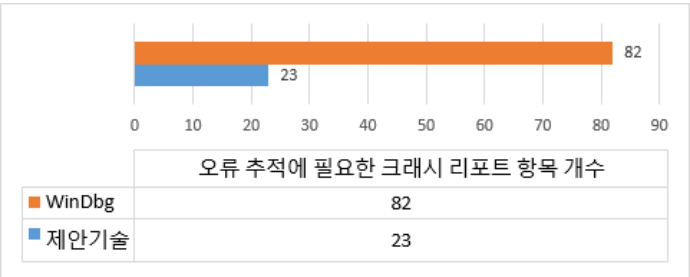


그림 3 WinDbg와 제안 기술의 오류 수집 데이터 비교

3.2 실험 결과

선정한 기존 오류 추적 도구인 WinDbg [5]는 그림 3과 같이 실행 환경 정보를 포함한 82개의 메모리 덤프 분석 결과를 출력한다. 이는 개발자의 오류 분석 난이도가 증가하여 오류 추적을 위한 비용이 많이

발생한다. 반면, 제안한 크래시 리포트는 예외 정보를 통해 발생한 오류의 종류를 분석하고, 타임스탬프를 이용하여 오류가 발생한 프로젝트의 모듈 및 소스 파일을 탐지하며, 오류가 발생한 모듈의 물리 주소를 분석하여 소스 코드의 라인 위치 식별이 가능한 결과를 보였다. 이는 WinDbg [5]와 같은 환경과 자원에서 약 72% 감소한 23개의 분석 결과만 확인해도 WinDbg [5]와 동일하게 성공적으로 오류 추적을 할 수 있는 것을 확인할 수 있었다. 따라서, 크래시 리포트의 크기가 경량화되고 분석 비용이 감소하여 품질이 향상했다. 또한, 실행하고 있는 모든 모듈 리스트나 메모리 정보를 모두 전송하여 개인 정보까지 전송될 수 있는 기존 방법과 달리 실행한 모듈에 대한 연관 정보와 예외 정보만 수집하기 때문에 개인 정보를 이용하지 않아 보안성까지 향상하는 것을 확인하였다.

4. 결론 및 향후 계획

본 논문에서는 소프트웨어의 디버깅 비용 감소와 효율성 증대를 위해 보안성을 고려한 고품질 경량 크래시 리포트 자동 생성 방법을 제안한다. 제안 기술은 이식성이 높은 라이브러리 형태로 소프트웨어에 쉽게 추가하여 배포할 수 있다. 배포된 소프트웨어가 크래시 발생을 인지하고, 오류 추적에 필요한 데이터만 수집해 크래시 리포트를 생성 및 전송한다. 이후 크래시 리포트를 전달받은 개발자가 핵심 오류 정보로 실행 환경에 독립적으로 오류 위치 추적이 가능한 상대적 가상 주소(Relative Virtual Address)와 오류가 발생한 모듈을 구분할 수 있는 타임스탬프를 활용하여 효율적인 디버깅 작업을 할 수 있다.

향후 계획으로는 제안 기법을 기반으로 스테이트먼트 단위의 결함 위치 자동 식별까지 할 수 있는 디버깅 연구를 진행할 것이다.

Acknowledge

이 논문은 2016년 정부(교육부)의 재원으로 한국연구재단-이공학 개인기초연구 지원 사업(No. 2016R1D1A1B03934610), 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학 지원사업(No.2015-0-00914)의 연구결과로 수행됨

참고 문헌

[1] Apple Crash Reporter, https://developer.apple.com/library/archive/technotes/tn2004/tn2123.html

[2] Microsoft Windows Error Reporting, https://docs.microsoft.com/en-us/windows/desktop/wer/about-wer

[3] K.Glerum, K.Kinshumann, S.Greenberg, G.Aul, V.Orgovan, G.Nichols, D.Grant, G.Loihle and G.Hunt, Debugging in the (very) large: ten years of implementation and experience, Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp. 103-116, 2009.

[4] Overview of memory dump file options for Windows, https://support.microsoft.com/en-us/help/254649/overview-of-memory-dump-file-options-for-windows

[5] Microsoft WinDbg, https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-tools

[6] Processhacker, https://github.com/processhacker/processhacker

[7] Juliet Test Suite for C/C++, https://samate.nist.gov/SRD/testsuite.php

생성적 적대 신경망을 이용한 소프트웨어 결함 예측

이주혜 ○, 진영인, 남재창

한동대학교 전산전자공학부

joohye97@naver.com, jinyi1187@gmail.com, jcnam@handong.edu

Software Defect Prediction using Generative Adversarial Networks

Joohye Lee ○, YoungIn Jin, Jaechang Nam

School of Computer Science and Electronic Engineering, Handong Global University

요 약

소프트웨어 공학에서 제한된 자원을 효율적으로 배분하고 소프트웨어의 품질을 보증하기 위해서 소프트웨어의 결함 모델 예측 기법이 활발히 연구되고 있다. 특히 기계학습을 활용한 소프트웨어 결함 예측 모델 기법이 많이 제안되었다. 하지만 기계학습 모델의 학습을 위해 주어진 데이터셋(dataset)의 결함 있는 데이터(buggy data)와 결함 없는 데이터(clean data)의 비율이 불균형하여 모델이 제대로 학습되지 못하는 결과가 있어왔고, 이로 인해 결함 예측 성능에 부정적인 영향을 주었다. 따라서 모델 학습 데이터의 결함 있는 데이터와 결함 없는 데이터의 불균형 문제를 해결하기 위해 딥러닝 기법인 생성적 적대 신경망을 활용할 수 있는지 검증해 보았다. 검증 결과에서 생성적 적대 신경망을 이용한 모델의 예측 성능이 기존 연구의 예측 성능들과 비교하였을 때 F-measure 나 AUC 등의 예측 척도 등에서 낮은 결과를 보여주었다. 이런 결과는 딥러닝 기반의 기법들은 많은 양의 훈련데이터가 있어야 한다는 점과, 생성적 적대 신경망의 적절하지 못한 활성화 함수와 은닉층의 적은 개수 생성적 적대 신경망 등이 잠재적인 이유로 판단된다. 이러한 검증 결과를 바탕으로 향후 불균형한 데이터 문제를 해결하여 결함 예측 성능을 향상시킬 수 있는 다양한 기법들을 연구할 계획이다.

1. 서 론

소프트웨어공학에서 소프트웨어 결함 예측 기술은 소프트웨어 품질 관리를 돕는 중요한 기술 중 하나이다[1]. 대형 소프트웨어의 경우, 배포하기 전에 수십만개의 파일들을 직접 검토하고 코드들을 테스트하는 것이 필수적이다. 이를 위해서는 수많은 인력과 막대한 시간이 필요하게 된다. 하지만 소프트웨어 결함 예측 기술을 이용한다면, 제한된 자원을 효율적으로 배분할 수 있을 뿐만 아니라 소프트웨어의 품질을 보증할 수 있다[1].

그리하여 많은 소프트웨어 공학 분야의 연구자들은 기계학습 기법을 활용하여 소프트웨어 결함 예측을 하려는 시도를 많이 해왔다[2]. 하지만 기계학습 모델의 학습을 위해 주어진 데이터셋의 결함 있는 데이터와 결함 없는 데이터의 불균형한 비율은 모델이 제대로 학습되지 못하는 결과를 낳았으며, 이로 인해 결함 예측 성능은 부정적인 영향을 받게 되었다[3]. 따라서 모델의 학습 데이터의 결함 있는 데이터의 비율과 결함 없는 데이터의 비율을 맞추는 것은 모델의 예측 성능 향상을 위해 매우 중요한 전처리 과정 중 하나이다[4].

기존에 이러한 학습 데이터의 불균형 문제를 해결하는데 보편적으로 사용되는 기법과 최근 기법들로 SMOTE(Synthetic Minority Oversampling Technique), SMOTUNED 와 같은 알고리즘과 BAGAN(Balancing GAN)이 있다 [5][6]. SMOTE, SMOTUNED 는 불균형한 클래스 데이터에서 소수 클래스의 데이터를 생성하여 그 균형을 맞춘다. BAGAN 은 이미지로 구성된 불균형한 학습데이터를 균등하게 만들어 기계학습 모델 분류기의 성능을 향상시켰다. 하지만 SMOTE, SMOTUNED 는 주어진 학습데이터만을 이용하여 새로운 데이터를 생성하기 때문에, 기존 데이터의 범위를 넘어가는 데이터가 주어졌을 때는 비교적 높은 성능을 보여주지 못한다는 점에서 한계가 있다. 또한 BAGAN 같은 경우, 이미지를 다룬다는 점에서 이미지가

아닌 결함 있는 데이터셋을 학습시키기에는 부적합하다고 판단된다.

따라서 본 연구의 목적은 소프트웨어 결함 예측을 위한 기계학습 모델 학습에 주어진 불균형한 학습 데이터셋으로 인한 예측 성능 문제를 해결하기 위해서 딥러닝 기법인 생성적 적대 신경망 (Generative Adversarial Networks, GAN)의 아이디어를 활용할 수 있는지 검증하는데 있다. GAN 은 경쟁적인 신경망 구조로, 학습을 위해 주어진 데이터의 분포를 학습하여 그와 비슷한 분포를 가진 가짜 데이터를 생성하는 알고리즘이다. 따라서 GAN 을 활용하면 실제 데이터와 비슷한 분포의 가짜 데이터를 생성할 수 있으므로 불균형한 학습 데이터의 균형을 맞출 수 있을 것이라 판단된다.

2. 선행 연구

2.1 SMOTE

기존 기계학습 모델의 학습 데이터의 불균형한 문제를 해결하기 위해서는 데이터의 개수가 소수인 클래스의 인스턴스 과대표집 (Over-Sampling), 다수인 클래스의 인스턴스 과소표집(Under-Sampling) 등의 해결 방법이 있다. 하지만 소수 클래스의 과대표집은 단순히 기존 소수 클래스의 데이터를 복사하여 그 개수의 비율을 맞추기 때문에, 모델이 학습데이터에는 잘 맞지만 일반성이 떨어지는 과대적합(overfitting) 문제가 생길 수 있다[7]. 또한 과소표집은 다수 클래스 인스턴스를 소수 클래스의 크기만큼 제거하므로, 두 클래스의 차이를 나타내 주는 중요한 인스턴스를 제거하는 문제를 낳을 수 있다[7]. 따라서 과대표집, 과소표집을 보완하는 방법으로 SMOTE 가 제안되었다. SMOTE 는 k-NN(k-Nearest Neighbor)알고리즘을 이용하여 새로운 소수 클래스의 인스턴스를 생성하는 방법이다. 여기서 k-NN 은 유클리디안 거리 계산법을 이용하여 거리를 측정하여 주어진 데이터를 k 개의 레이블을 참조하여 분류하는 알고리즘이다[3]. SMOTE 는 기존의 소수 클래스의 데이터를 단순 복사하는 형식이 아니라 전체적인 학습 데이터의 특징을 반영하여 새로운 데이터를 만들기 때문에, 과도한 과대적합 없이 주어진 학습데이터의 균형을 맞추어 준다는 점에서 일반 과대표집보다 더 효율적으로 보인다. 하지만 SMOTE 또한 주어진 학습데이터를 이용하여 새로운 데이터를 만들기 때문에 여전히 과대적합의 문제는 존재한다.

2.2 Balancing GAN

SMOTE 가 주어진 학습데이터만을 이용하여 새로운 데이터를 만들어내는 방식이었다면, BAGAN 은 주어진 학습데이터의 분포를 학습하여, 학습데이터의 분포와 유사한 완전 새로운 데이터를 생성한다[6]. BAGAN 은 Balancing Generative Adversarial Networks 으로, GAN 과 오토인코더 (Autoencoder)의 개념을 함께 이용한다. GAN 은 적대적이고 경쟁적인 신경망 모델로, 입력된 학습 데이터의 분포를 생성자(generator)가 학습하고 식별자 (discriminator)는 생성자가 생성한 데이터가 원본 데이터와 유사한지 판단하며 생성자에 피드백을 하여 식별 성능을 높인다. 오토인코더는 주어진 데이터에서 특징(feature)을 추출하여 그 특징을 이용하여 원본과 유사한 데이터를 만드는 기법이다. BAGAN 은 이를 이용하여 불균형한 클래스의 균형을 맞추었고, 모델의 성능을 향상시켰다[6]. 하지만 BAGAN 은 주어진 학습데이터가 이미지라는 점에서, 본 연구가 적용하고자 하는 메트릭(metric)을 적용하지 못한다는 한계점이 있다.

3. 방법의 설계 및 구현

3.1 적용 개념

본 연구는 GAN 을 이용하여, 소프트웨어 결함 예측 모델의 학습을 위해 주어진 데이터의 결함 있는 데이터와 결함 없는 데이터의 개수의 균형을 1:1 로 맞추는 것을 목표로 한다.

GAN 은 신경망 이론을 이용한 생성모델로, 기본적으로 생성자와 식별자로 이루어져 있다. 생성자와 식별자는 경쟁적인 신경 네트워크 시스템에 의해 구현되어 있어, 서로 경쟁하며 각자의 성능을 점차 개선해 나간다. 생성자는 주어진 데이터의 분포를 알아내서 그와 최대한 비슷한 데이터 분포를 생성하는 것을 목표로 한다. 식별자는 현재 자신이 갖고 있는 데이터가 학습용 데이터로부터 온 것인지, 생성자가 생성한 데이터인지를 분류하는 것을 목표로 한다. 분류는 각각의 경우에 대한 0~1 사이의 확률값을 산출하는 것인데, 1 에 가까울 수록 자신이 구별한 데이터가 학습용 데이터와 가깝다는 것이고 반대로 0 에 가까울 수록 생성자가 생성한 데이터와 가깝다는 것을 의미한다. 학습이 이뤄지기 전에는 학습용 데이터의 분포와 생성자가 생성한 데이터 분포는 일치하지 않는다. 또한 식별자의 성능 분포 또한 원데이터와 생성된 데이터를 잘 구분할 수 있을 정도로 고르지 않다. 학습을 시작하게 되면 성능이 향상되는데,

이때 학습은 최적의 가중치값(weight)과 편향값(bias)을 찾기 위해 생성자와 식별자 각각의 손실함수(loss function)를 최소화하는 방향으로 진행된다. 처음에 가중치는 무작위로 초기화 되지만, 학습이 진행되면서 손실함수 값이 역전파를 통해 각 층의 가중치에 전달된다. 이 가중치는 성능 향상의 방향과 크기를 결정한다. 생성자와 식별자의 경쟁을 통해 손실함수가 업데이트 되면서 각각의 성능 또한 향상된다. 학습용 데이터를 Pdata(x), 생성된 데이터를 Pg(x)라고 했을 때, 식별자는

$$D(x) = \frac{Pdata(x)}{Pdata(x)+Pg(x)}$$

를 이용하여 자신의 성능을 향상시킨다. 학습이 진행되면서 생성자가 생성한 데이터와 학습용 데이터의 분포는 점점 일치해지며, 궁극적으로 Pdata(x)와 Pg(x)가 일치해져 식별자의 분포는 두 분포를 전혀 구분하지 못하는 D(x) = ½ 상태가 된다[7].

3.2 구현

본 연구에서는 기존의 이미지 처리를 하는 GAN 을 개조하여 사용하여 메트릭 값을 생성해 낼 수 있는 GAN 을 구현한다. 기존 GAN 과 다른 점은 입력과 출력 데이터의 타입이 다르다는 것과, GAN 내부에서 실행되는 신경망에서의 활성화 함수(activation function)가 기존 GAN 과 다른 활성화 함수 사용하여 데이터를 생성해낸다는 데에 차별점을 두고 있다. 활성화 함수란, 입력 신호를 출력 신호로 변환하여 활성화의 여부를 결정할 수 있게 하는 역할을 한다. 일반적으로 이미지를 처리하는 GAN 의 생성자 부분은 문제의 복잡도에 따라 은닉층(hidden layer)의 개수를 다르게 설정한다. 그리고 이 각각의 층의 가중치와 편향값에 sigmoid, ReLu, tanh(hyperbolic tangent) 같은 활성화 함수를 적용하여 최종 결과값을 산출한다. 이때 이미지의 픽셀값의 범위와 맞는 결과값을 만들기 위해 활성화 함수로 주로 sigmoid 나 tanh 와 같은 함수들이 이용된다. 식별자도 생성자와 마찬가지로 은닉층을 설정하고 이에 ReLu, Softmax 와 같은 활성화 함수를 적용한다. 이때, 식별자는 생성자의 결과와 학습데이터의 값의 분포가 얼마나 다른 지를 0~1 사이의 확률값으로 표현하여야 하므로, sigmoid 함수를 적용한다.

GAN 은 이미지 처리 영역에서 기존에 존재하던 딥러닝 기법과 비교하여 우월한 성능 향상을 보였기에 대게 이미지 생성과 관련되어 활용되었다. 반면에, 메트릭과 같은 값의 생성에 관해서는 거의 활용되지 않았었다. 따라서 본 연구는 이미지가 아닌 메트릭 값의 처리를 위해 기존의 이미지 처리를 위한 GAN 과 몇 가지 영역에서 차별점을 두었다. 먼저 생성자 부분에서 기존의 GAN 은 이미지의 픽셀을 처리하기 위해 활성화 함수로 sigmoid 나 tanh 를 주로 활용하였는데, 이는 픽셀값이 0 에서 1 혹은 -1 에서 1 사이의 값이기 때문이다. 하지만 메트릭는 이미지의 픽셀값이 아닌 이상 0 에서 1 사이의 값을 산출하지 않아도 되기 때문에 활성화 함수로 sigmoid 나 tanh 를 적용하지 않았다.

또한 다양한 크기의 메트릭이 GAN 의 생성자와 식별자의 입력값으로 주어지게 된다. 이때 입력값은 결과값 산출을 위하여 GAN 의 은닉층과 출력층을 거치며 가중치값과 편향값과의 행렬 연산을 한다. 이를 위해서 생성자와 식별자의 가중치의 행렬의 모양은 입력 메트릭의 행렬 모양에 따라 그 크기가 동적으로 조정된다.

표 1. 해당 실험에 사용된 데이터셋[9]

Subject	# of files(instances)	# of buggy files(%)	# of metrics
Equinox(EQ)	325	129(39.69%)	61
Eclipse JDT core(JDT)	997	206(20.66%)	61
Apache Lucene(LC)	399	691(9.26%)	61
Mylyn(ML)	1862	245(13.16%)	61
Eclipse PDE UI(PDE)	1492	209(14.01%)	61

4. 실험 및 평가

본 연구는 개조된 GAN 으로 생성된 데이터와 기존의 데이터로 소프트웨어 결함 예측 성능을 평가하기 위하여 Weka 를 활용하였다[10]. Weka 는 자바로 개발된 오픈소스 기계학습 데이터 마이닝 툴이다. 해당 실험을 위하여 주어진 데이터는 Transfer Defect Learning 의 실험을 위하여 사용되었던 데이터들과 동일하다[9]. 주어진 데이터셋의 정보는 다음과 같다.

표 1 에서 보는 것과 같이 해당 데이터집합들은 일반적으로 결함 있는 데이터의 비율이 결함 없는 데이터의 비율보다 작다. 따라서 모델의 예측 성능 향상을 위하여 결함 있는 데이터의 비율이 결함 없는 데이터와 1:1 비율이 되도록 개조된 GAN 을 통해 데이터를 생성한다. 해당 실험에서 생성된 데이터로 모델의 예측 성능을 측정하기 위해서는 다음과 같은 과정을 거친다.

먼저 주어진 데이터셋을 학습용 데이터와 테스트 데이터로 나눈다. 그 다음에 추출한 학습용 데이터에서 결함 있는 데이터와 결함 없는 데이터의 비율을 확인하고 결함 없는 데이터만을 다시 따로 추출한다. 이는 일반적으로 결함 없는 데이터의 비율이 결함 있는 데이터의 비율보다 작기 때문에, 결함 없는 데이터가 결함 있는 데이터와 1:1 비율이 되도록 생성하는 것을 기본 조건으로 설정하였기 때문이다. 따라서 따로 추출한 결함 있는 데이터를 개조한 GAN 에 넣어 새로운 데이터를 생성하면, 결함 없는 데이터와의 비율이 맞게 된다. 이렇게 새로 생성된 데이터와 기존 데이터로 모델의 예측 성능을 평가하기 위해 두 데이터가 합쳐진 데이터셋을 이용하여 예측 모델을 학습하고, 앞서 추출한 테스트 데이터로 학습된 예측 모델을 평가하여 측 성능을 확인하였다[10]. 기계학습 모델로는 로지스틱 회귀분석 (Logistic Regression) 알고리즘을 사용했다.

표 2. 각 모델들의 예측 성능 비교 분석표

Project	Precision			Recall			F-measure			AUC		
	General	SMOTE	GAN	General	SMOTE	GAN	General	SMOTE	GAN	General	SMOTE	GAN
ML	**0.848**	0.311	0.404*	**0.882**	0.602*	0.091	**0.859**	0.401*	0.148	**0.735**	0.762*	0.267
EQ	**0.628**	0.619*	0.466	**0.626**	0.567*	0.419	**0.622**	0.586*	0.437	**0.608**	0.588*	0.461
JDT	**0.783**	0.531	0.476	**0.758**	0.651*	0.258	**0.767**	0.583*	0.309	**0.786**	0.811*	0.447
PDE	**0.835**	0.293	0.597*	**0.850**	0.449*	0.112	**0.841**	0.354*	0.175	**0.616**	0.646*	0.405
LC	**0.875**	0.235	0.685*	**0.856**	0.481*	0.120	**0.865**	0.315*	0.184	**0.598**	0.683*	0.445

위의 표 2 은 GAN, SMOTE 그리고 아무것도 적용하지 않았을 때(General) 새로운 데이터를 생성하여 기존의 데이터와 합친 후 모델에 적용한 결과값을 보여준다. 해당 실험은 통계학적 가설 검증 실험을 위해 윌콕슨 부호 순위 검정(Wilcoxon signed-rank test, $p<0.05$)를 활용하였다. 굵은 글씨는 p 값이 0.05 보다 작은 경우, GAN 과 General 을 비교하였을 때 성능이 좋은 모델을 표시하였으며, 마찬가지로 별표(*)는 p 의 값이 0.05 보다 작은 경우 GAN 과 SMOTE 모델을 비교하였을 때 성능이 더 좋은 모델을 표시하였다. SMOTE 와 General 사이에서 더 높은 성능 결과는 밑줄로 표시하였다. GAN 이 다른 모델들에 비해 성능이 뛰어난 경우는 Precision 에서 ML 과 PDE, LC 프로젝트에 대해 SMOTE 와 비교하였을 때 뿐이다. 다른 경우에는 아무것도 적용하지 않았을 때(General)와 SMOTE 를 적용하였을 때가 성능이 더 우수하다. AUC 결과는 SMOTE 가 대체적으로 제일 좋은 성능을 보여주었다.

5. 결론

GAN 을 이용한 모델의 예측 성능은 SMOTE 와 아무것도 적용하지 않은 모델의 것과 비교했을 때 현저하게 낮다. 이는 다음과 같은 원인으로 인한 것이라고 추측해볼 수 있다. 먼저 신경망 이론에서 모델의 예측 성능을 향상시키는 데 중요한 역할을 하는 가중치를 다루는 은닉층의 개수가 모델을 학습시키기에 충분하지 않았다고 판단된다. 은닉층 안의 가중치는 예측값을 가장 잘 맞추도록 조정되어 모델의 성능에

큰 영향을 끼친다. 따라서 은닉층을 적절한 개수를 설정하는 것이 필요하다. 하지만 본 연구에서는 은닉층의 개수가 기존 GAN 의 이미지 생성에 적합한 개수로 설정되어 있으므로, 메트릭을 다루는 데 충분하다고 판단할 수 없다. 다음으로는 모델의 학습을 위해 주어진 데이터의 개수가 충분하지 않았다. 즉 모델이 주어진 데이터에 대한 학습을 하기 위해서는 충분한 학습 데이터가 주어져야 하는데, 본 연구의 실험에서 사용된 학습데이터의 개수는 충분하지 않다는 것이다. 한 가지 예로, 기계학습에서 가장 많이 이용되는 예제인 MNIST 의 경우, 모델의 학습을 위해 대략 50,000~60,000 개의 데이터가 사용된다. 하지만 본 연구의 실험 데이터는 대략 300~1,800 개로, MNIST 와 비교하였을 때 현저하게 데이터가 부족하다고 판단할 수 있다. 이로 인해 모델의 학습이 충분히 이루어지지 못하였고, 결과적으로 모델의 예측 성능에 부정적인 영향을 주었다고 판단된다.

소프트웨어의 결함을 예측하는 것은 중요하다. 따라서 결함 예측 모델에 주어진 데이터셋에서 결함 있는 데이터와 결함 없는 데이터의 비율을 맞추는 것은 결함 예측 기술의 중요한 과정 중 하나이다. 향후에는 모델 학습을 위해 주어진 데이터셋의 비율을 맞추기 위해서, 적절한 은닉층의 개수를 찾는 것과 충분한 개수의 데이터셋으로 모델을 학습하는 것이 필요해 보인다. 부수적으로 지금보다 더 적합한 활성화 함수를 적용한다면, GAN 을 이용하여 데이터셋의 비율을 맞춰 모델에 적용하였을 때, 기존의 다른 예측값들보다 더 높은 예측 성능을 보일 것으로 기대된다.

※ 본 연구는 과학기술정보통신부와 정보통신기술 진흥센터의 2019 년도 소프트웨어 중심대학 지원사업(2017-0-00130)과 2018 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2018R1C1B6001919).

참고문헌

[1] Güneş Koru, A & Tian, Jeff. An empirical comparison and characterization of high defect and high complexity

modules. Journal of Systems and Software. 67. 153-163. 2003

[2] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan and Fatima Alsarayrah, "Software Bug Prediction using Machine Learning Approach", International Journal of Advanced Computer Science and Applications(IJACSA), 9(2), 2018

[3] Chawla, Nitesh & Bowyer, Kevin & O. Hall, Lawrence & Philip Kegelmeyer, W. "SMOTE: Synthetic Minority Over-sampling Technique". J. Artif. Intell. Res. (JAIR), 2002

[4] Pears, R., Finlay, J. & Connor, A.M. "Synthetic Minority Over-sampling TEchnique (SMOTE) for Predicting Software Build Outcomes", Proceedings of the Twenty-Sixth International Conference on Software Engineering and Knowledge Engineering (SEKE), 2014

[5] Longadge, Rushi and Snehalata Dongre. "Class Imbalance Problem in Data Mining Review." CoRR abs/1305.1707, 2013

[6] Mariani, Giovanni, Florian Scheidegger, Roxana Istrate, Constantine Bekas and A. Cristiano I. Malossi. "BAGAN: Data Augmentation with Balancing GAN" CoRRabs/1803.09655, 2018

[7] Kotsiantis, Sotiris B., Dimitris Kanellopoulos and Panayiotis E. Pintelas. "Handling imbalanced datasets : A review.", GESTS International Transactions on Computer Science and Engineering, Vol.30, 2006.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative adversarial nets", Advances in neural information processing systems, 2672-2680, 2014

[9] Jaechang Nam, Sinno Jialin Pan and Sunghun Kim, "Transfer Defect Learning",*in Proceedings of 35th International Conference on Software Engineering (ICSE 2013)*, San Francisco, May 18-26, 2013

[10] https://www.cs.waikato.ac.nz/ml/weka/

사이버-피지컬 시스템의 상호운용성 상충관계 분석에 관한 연구

김의섭 O, 유준범
건국대학교 컴퓨터공학과
atang34@konkuk.ac.kr, jbyoo@konkuk.ac.kr

A Study on Interoperability Conflict Analysis for Cyber-Physical System

Eui-Sub Kim O, Junbeom Yoo
Division of Computer Science and Engineering

요 약

사이버-피지컬 시스템은 다양한 이종의 어플리케이션 및 시스템이 동적으로 결합되어 동작하는 개방형(open-ended) 시스템이다. 하지만 상충되는 목적을 가진 어플리케이션 및 시스템이 결합된다면 상호운용성을 저해할 수 있다. 상호운용성의 실패는 시스템의 예측할 수 없는 동작을 발생시키고, 이는 시스템의 기능적 실패뿐만 아니라 사고로 이어지기 때문에 상호운용성의 상충여부를 개발 초기에 분석할 필요가 있다. 본 논문에서는 요구사항 단계에서 상호운용성의 상충을 분석할 수 있는 방안과 이를 명세할 수 있는 SysML의 확장 형태를 제시한다. 이를 통해 상호운용성의 상충 여부를 미연에 식별하여 대처할 수 있을 것으로 기대 할 수 있다.

1. 서 론

사이버-피지컬 시스템(CPS: Cyber-Physical System)은 피지컬 컴포넌트와 사이버 컴포넌트가 네트워크로 밀접하게 연결된 시스템으로서, 사이버 컴포넌트가 피지컬 컴포넌트를 모니터링하고, 연산하여 피드백 루프를 통해 피지컬 컴포넌트를 제어하는 시스템이다. 현재 안전 필수 도메인을 비롯한 다양한 도메인(지능적 교통량 관리 및 스마트 공장, 헬스 케어, 스마트 그리드 등)에 적용되고 있으며, 이런 사이버-피지컬 시스템은 기존 시스템과 다른 다양한 지능적 기능을 제공함으로써 삶의 질을 높이거나 기술적 발전을 이루는데 앞장서고 있다[1].

사이버-피지컬 시스템은 다양하게 변화하는 실세계에서 동작하는 개방형(open-ended) 시스템이며[2] 이종(heterogeneous)의 시스템이 커뮤니케이션하는 시스템이다. 따라서 동일 또는 이기종의 시스템 및 제품, 서비스가 특별한 제약이 없이 서로 호환되어 사용될 수 있도록 상호운용성(Interoperability)이 보장되어야 하며[3], 새로운 시스템 및 제품, 서비스가 추가되더라도 기능적 저하가 발생하지 않아야 한다. 또한, 동적이고 불확실한 현실 세계의 다양한 상황에 대해서도 충분히 인식하여 대처 및 대응할 수 있는 능력이 요구되고 있다[4].

하지만 사이버-피지컬 시스템의 이종성과 동적인 환경, 다양한 시스템 및 제품, 서비스의 결합, 새로운 기능의 추가 등은 사이버-피지컬 시스템의 상호운용성 보장을 저해하는 요소이다. 특히 상충되는 목적을 가진 시스템 및 제품, 서비스가 결합되면, 사이버-피지컬 시스템이 어떤 동작을 할지 예측할 수 없게 되는 문제가 발생한다. 이런 문제는 단순하게 시스템의 성능 저하뿐만 아니라 안전을 위협하는 사고로 이어지게 되기 때문에 반드시 개발 및 설계 단계에서 충분히 분석하고 해결해야 할 문제이다.

본 논문에서는 사이버-피지컬 시스템의 상호운영성의 내재된 상충 문제를 해결하고자 상호운용성의 충돌을 유발하는 상충관계 분석 방안을 제시한다. 제시하는 방안은 요구사항 분석 단계에서 기능적 요구사항과 함께 상충관계를 분석해 개발 초기에 상충관계를 식별하여 상호운용성을 확보할 수 있도록 지원한다. 이를 통해 전체 개발 비용을 절감하고 보다 안전한 시스템의 개발이 가능할 것이다. 이를 위해 상충관계에 대한 분류 및 분석 방법을 제시하고 이를 명세할 수 있도록 SysML[5]의 확장된 프로파일을 제시한다.

상호운용성의 상충관계를 요구사항 분석 단계에서 수행함으로써 상호운용성의 상충 여부를 미연에 식별하여 대처할 수 있도록 시스템 설계가 가능할 것으로 예상되며, 또한 새로운 시스템과의 결합을 가능하도록 유연하고 확장성 있는 시스템의 개발에 도움이 될 것으로 기대할 수 있다.

2. 사이버-피지컬 시스템의 상호운용성 상충관계

상호운용성이란 시스템 및 제품, 서비스 가 추가적인 노력 없이도 타 시스템 및 제품, 서비스와 함께 동작할 수 있는 능력이다[4]. 사이버-피지컬 시스템은 다양한 이종의 시스템 및 제품, 서비스가 커뮤니케이션 하는 시스템으로서 상호운용성의 보장이 중요한 시스템이다. 시스템 내부적으로는 새로운 서비스 및 어플리케이션이 추가되어 상호운용 될 수 있으며 시스템 외부적으로는 다른 사이버-피지컬 시스템과 상호운용 될 수 있다. 그림 1은 사이버-피지컬 시스템의 아키텍처와 상호운용되는 요소를 나타낸 그림이다.

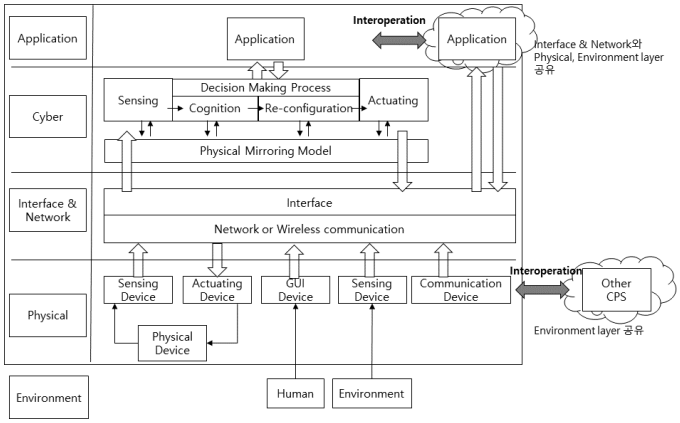


그림 1 사이버-피지컬 시스템의 아키텍처 및 상호운용성

상호운용성의 상충이란 상호 운용되어야 하는 어플리케이션 및 시스템이 제대로 커뮤니케이션을 하지 못해 상호 운용되지 못하는 정적인 상충(구조적 연결성 상충)과 서로 다른 어플리케이션 및 시스템이 각자의 연산 결과가 불일치하여 어떤 것을 수행해야 할지 판단하기 어려운 상황인 동적인 상충(기능적 상충)을 의미한다.

정적인 상충은 사이버 컴포넌트와 피지컬 컴포넌트의 상호운용 뿐만 아니라 이종의 서비스 및 어플리케이션, 이종의 네트워크 프로토콜, 제조업체가 다른 디바이스 및 어셋, 다른 사이버-피지컬 시스템과의 상호운용 등 다양한 곳에서 존재한다. 뿐만 아니라, 운용 중 다른 시스템 및 제품, 서비스와 동적인 조합, 유지보수를 위한 제품의 교체, 제품 및 어플리케이션의 업데이트 등에 의해서 정적 상충 상황이 발생한다. 해당 상충을 해결하기 위해서는 RAMI 4.0 표준[6]이나 AUTOSAR 표준[7]과 같은 산업 표준을 준수하여 개발하여 커뮤니케이션 측면에서 상충 상황이 발생하지 않도록 대비해야 한다.

반면, 동적인 상충은 기능적 상호운용성 상충을 의미한다. 정적인 상충은 참조 모델 및 표준 준수, 인터페이스 확인 등으로 보장 가능한 반면 동적인 상충은 분석 및 식별하기가 보다 어렵다. 예를 들어 고속도로의 사고를 감지하여 전광판 출력을 통해 사고를 대처하는 시스템에 새롭게 교통량을 확인하여 전광판을 통해 차로를 제어하는 서비스를 추가할 경우 두 서비스간 기능적 상호운용성 상충이 발생하여 사고 감지 및 대처 시스템은 전광판에 진입 통제 명령을 내리지만, 교통량 제어 시스템은 전광판에 진입 허용 명령을 내리는 상충상황이 발생할 수 있다. 문제는 이런 상충상황이 동적이고 무수히 많은 불확실성이 존재하는 실세계에서 모든 상황이 통제되도록 설계하는 것이 어렵다는 것이다. 따라서 이를 개발 단계 초기에 분석하여 대처할 수 있는 적절한 분석 방안이 필요하다.

- 정적 상충: 상호 운용되어야 하는 서로 다른 시스템 및 어플리케이션, 서비스가 커뮤니케이션에 실패하는 경우
- 원인: 데이터 포맷의 불일치, 통신 프로토콜의 불일치, 커뮤니케이션 지연 등
- 해결 방안: 데이터 포맷의 표준 확립과 준수, 커뮤니케이션을 위한 미들웨어 및 랩퍼 활용, RAMI 4.0 또는 AUTOSAR와 같은 참조 모델 및 표준 활용 등
- 동적 상충: 상호 운용되어야 하는 서로 다른 시스템 및 어플리케이션, 서비스의 판단 불일치로 어떤 판단을 따라 하는지 결정하지 못하는 경우
- 원인: 커뮤니케이션 지연, 동일한 환경에 대한 불일치 인지, 기능적 상충 등
- 해결 방안: 시나리오 기반 기능 분석을 통한 충돌 여부 평가

3. 사이버-피지컬 시스템의 상호운용성 상충관계 분석 방안

본 논문에서는 상호운용성의 상충관계를 식별하기 위해 요구사항 분석 단계에서 기능적 요구사항과 함께 상호운용성의 상충관계를 함께 분석하는 방법을 제시한다. 또한, 요구사항 명세와 상호운용성의 상충관계를 명세 및 분석할 수 있도록 확장된 형태의 SysML 프로파일을 제시한다. 확장된 SysML은 그림 2와 같다.

- requirement: requirement에는 어플리케이션 및 시스템에 요구되는 기능적 요구사항을 작성한다. requirement의 ID는 requirement를 식별할 수 있는 유일한 속성의 식별자이고, 이를 통해 추적성을 확보할 수 있다. Text는 requirement의 설명을 자연어로 기술하는 부분이다. Source는 requirement가 요구되는 요소의 이름을 기술하고, Source Layer는 그림 1의 아키텍처에 대응되는 Source의 Layer를 기술한다. Target은 명령을 내리고자 하는 요소의 이름을

기술하고, Target Layer는 그림 1에 대응하는 Target의 Layer를 기술한다. Command는 Target에 요구되는 명령을 기술하는 곳으로써 간략하고 명료하게 작성한다. Decision resource는 Source가 Command를 생성하는데 사용한 resource를 기술한다.

- conflict requirement: conflict requirement는 두 개의 어플리케이션 및 시스템이 동일한 Target 시스템에 다른 명령을 내리는 경우 Target의 대처 및 대응 방안에 대한 요구사항이다. Target이 동일하고 Source와 Command가 동일하지 않은지를 분석하여 체계적으로 도출이 가능하다. 또한, Source가 동일하더라도 Decision resource가 다를 경우 역시 충돌 가능성이 있으므로 해당 requirement를 통해 대처 방안을 분석해야 한다. 대처 방안으로는 요구사항 우선순위를 통해 대처할 수 있고, 상황에 따라 우선순위를 동적으로 할당해 대처할 수 있다.

- failsafe requirement: Conflict requirement가 수행되기 어려울 경우 대안으로 만족시킬 추가적인 요구사항을 의미한다. 상호운용성의 상충이란 시스템이 설계 동안에 예상하지 못한 상황에 도달했다는 것을 의미한다. 실세계에는 다양한 상황이 발생하기 때문에 분석가가 모든 것을 식별하는데 한계가 있다. 경우에 따라서 상호운용성의 상충 상황은 Conflict requirement도 온전히 수행되기 어려운 상황일 가능성이 크다. 따라서 시스템 롤백 또는 초기화 등의 기능을 통해 시스템이 지속적으로 동작 가능하도록 기능을 분석할 필요가 있고, failsafe requirement는 이런 상황에 대한 시스템의 기능을 명세 한다.

그림 3은 제시한 방법 및 표기법을 바탕으로 실시간 돌발상황 감지 시스템 (AIDS, Automatic Incident Detection System)과 차로 제어 시스템 (LCS, Lane Control System)에 적용한 간략한 예제를 보여준다. 그림에서 AIDS 시스템의 요구사항 중 "사고 발생시 전광판에 진입 금지 출력" 요구사항과 LCS의 "고속도로 교통량에 따른 적절한 차선 유도" 요구사항이 전광판이라는 동일한 요소를 Target으로 삼는 것을 확인할 수 있다. 이 경우 Target에 상충된 Command가 발생할 수 있기 때문에, 이런 상충된 Command에 대해 어떻게 대처할 것인지 전략을 세울 필요가 있다. 예제에서는 우선순위를 설정하여 이를 해결하고자 하는 것을 확인할 수 있다. 이를 통해 기능적 요구사항뿐만 아니라 상충된 관계를 개발 초기인 요구사항 분석 단계에서 수행할 수 있고, 상충 가능한 부분을 체계적으로 식별해 적절한 대처 방안을 분석하는 것이 가능하다.

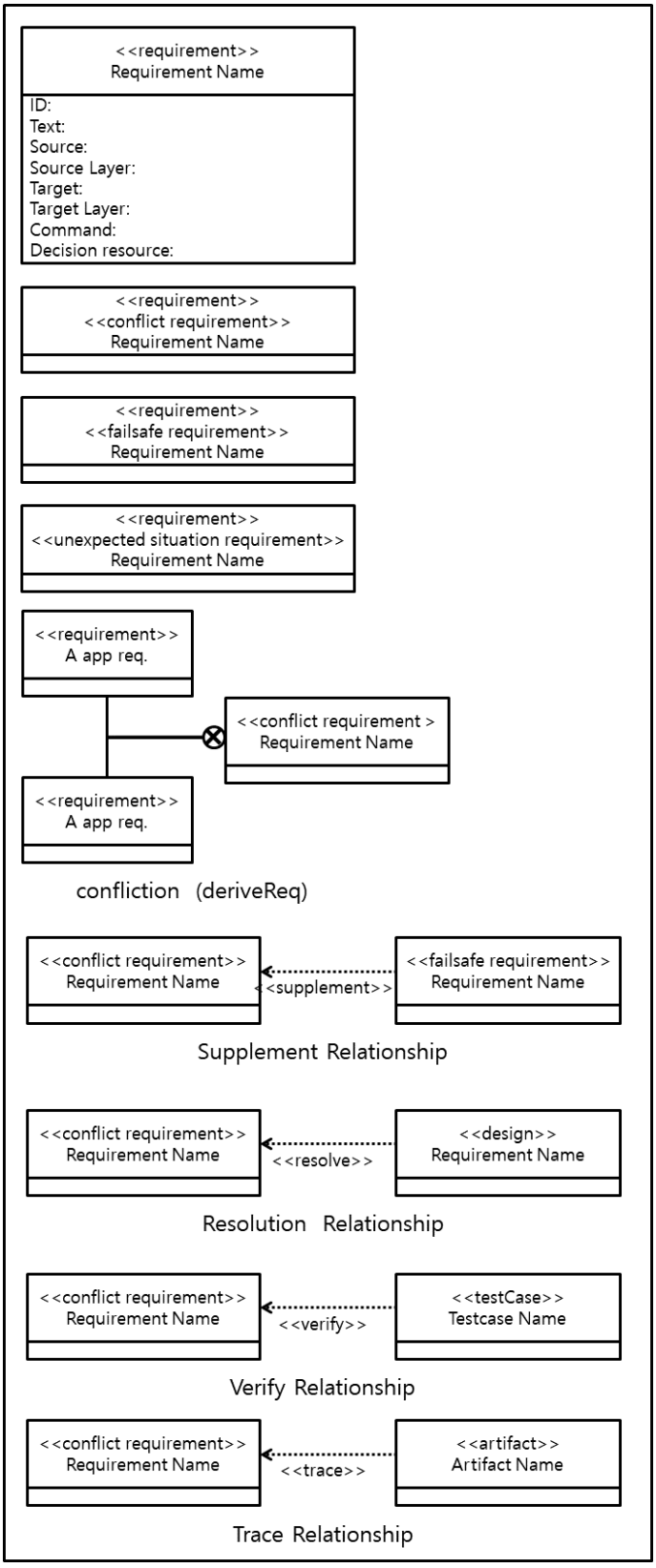


그림 2 상호운용성의 상충관계를 위한 확장된 SysML

제시한 방안의 문제점은 해당 분석을 요구사항 단계에서 수행하는 것이기 때문에 Target 또는 Command에 대한 명확한 식별 및 정의가 부족할 수 있다. 이 경우 개념적으로 정의를 하고 추후 명확한

요소가 정의되었을 때 적절하게 치환될 수 있도록, 사용한 용어 및 개념을 일관되게 유지하는 것이 중요하다. 또한, 모든 동일한 Target 마다 Conflict requirement를 작성하는 것은 지루하고 번거로운 일이 될 것이다. 이를 위해 앞으로 해당 분석을 자동/반자동으로 진행해 다이어그램을 생성해 주는 도구를 개발할 계획을 가지고 있다.

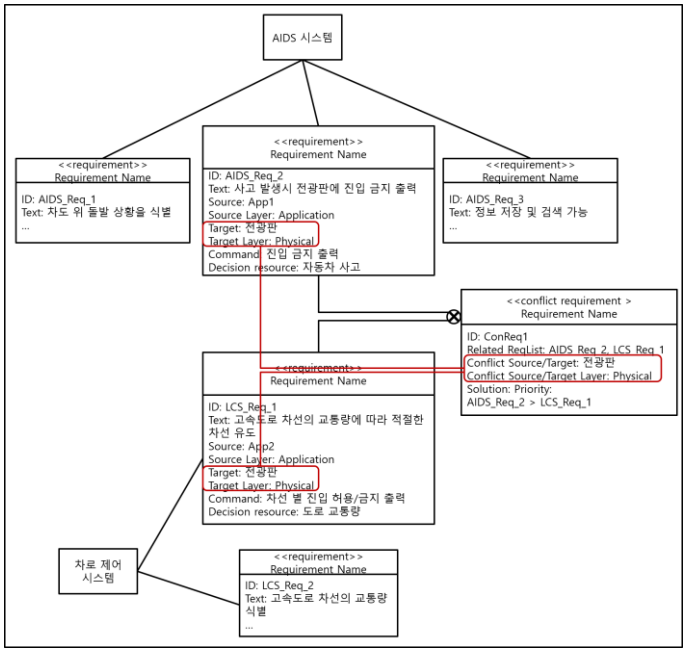


그림 3 상호운용성 상충관계를 기술한 요구사항 모델

4. 결 론

본 논문은 사이버-피지컬 시스템의 상호운용성의 상충관계를 요구사항 단계에서 분석할 수 있는 방안에 대해 제시한다. 요구사항 분석 단계에서 기능적 요구사항과 함께 상충관계에 대한 분석 및 명세를 수행할 수 있도록 SysML의 확장 형태를 제시하고, conflict requirement를 체계적으로 도출할 수 있는 방안에 대해서 제시했다. 설계의 이른 단계에서 해당 사항을 분석함으로써 개발 비용 절감 및 보다 안전하고 정확한 시스템의 개발이 가능할 것이라 기대할 수 있을 것이다. 추후, 해당 분석 모델을 바탕으로 자동으로 conflict requirement를 생성해주는 연구 및 도구 개발, 개발 사이클 전체를 통해서 해당 요구사항을 추적할 수 있는 방안에 대해서도 연구를 진행할 계획이다.

사 사

이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업 (NRF-2017M3C4A7066479)의 지원을 받아 수행한 연구임.

참고문헌

[1] Griffor, Edward R., et al. "Framework for Cyber-Physical Systems: Volume 1, Overview," Special Publication (NIST SP)-1500-201. 2017.

[2] Kit, Michal, et al. "An architecture framework for experimentations with self-adaptive cyber-physical systems." Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE Press, 2015.

[3] IEEE, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, 1990.

[4] Muccini, Henry, Mohammad Sharaf, and Danny Weyns. "Self-adaptation for cyber-physical systems: a systematic literature review." Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems. ACM, 2016.

[5] SysML, OMG Systems Engineering Domain Special Interest Group, http://syseng.omg.org/SysML.htm

[6] Status Report: Reference Architecture Model Industrie 4.0 (RAMI4.0), VDI/VDE Society Measurement and Automatic Control, July 2015

[7] Official website of the AUTOSAR Partnership: www.autosar.org

어휘적 특징에 기반한 프로그램 자동 패치 검색 공간 축소

안가빈 ○ 유신

한국과학기술원

agb94@kaist.ac.kr, shin.yoo@kaist.ac.kr

Search Space Reduction for Automated Program Repair Using Lexical Features

Gabin An ○ Shin Yoo

KAIST

Abstract

The Generate-and-Validate (G&V) technique is one of the main categories of Automated Program Repair (APR). Many G&V methods rely upon Plastic Surgery Hypothesis, also known as Redundancy Assumption, which assumes that the faulty program already contains the repair ingredients needed to synthesize a correct patch. Based on this assumption, the majority of techniques employ replacement or insertion operator that requires a specific code snippet to be taken from another location in the program. However, these techniques suffer from the search space explosion due to the large program size. In this paper, we propose an efficient ingredient screening procedure for one-line replacement operator by using only lexical features of source code. We investigate more than 50K real-world patches and report that the correct ingredients share several lexical features. Based on the observation, we establish an ingredient screening procedure that achieves 90% of search space reduction while showing 99% of recall on the CodRep corpus.

1. Introduction

Debugging is the process of identifying and removing faults within the software to prevent existing or potential failure of systems. However, as the size and complexity of software increase, debugging becomes much more difficult and time-consuming. Consequently, various Automated Program Repair (APR) [1-2] techniques have been proposed to fix faulty programs, without requiring developers' manual efforts, by utilizing modern computing power. More specifically, Generate-and-Validate APR [3-4], one of the main categories of APR, localizes a fault, generates candidate patches and evaluates the patched programs, typically through testing. Most of the existing work is based on the redundancy assumption [5-6] which claims that the repair ingredients already resides within the faulty program itself. To increase both effectiveness and efficiency of redundancy-based repair techniques, researchers have focused on analyzing the search space of repair ingredients and prioritizing them given a faulty code [7-8]. However, there has not been much work on screening out the unpromising ingredients in advance. If we can effectively reduce the ingredient search space, we can apply more intensive and even expensive prioritization techniques for the remaining ingredient candidates. So, in this paper, we propose a simple yet effective screening procedure that can substantially reduce the search space of repair ingredients. We systematically investigated one-line replacement patches, which replace a line with another line, from real-world Java programs [9]. Since a typical replacement operator substitutes a single modification point, usually a line or a statement, with another element at the same granularity level, we may apply this screening procedure to filter out the repair ingredients whenever we use a general replacement operator employed in many APR techniques, such as GenProg [3]. In the investigation, we analyze the relationship between the deleted line and the added line assuming that the deleted is the faulty line and the added is the correct line that fixes the fault. Based on this assumption, we report several conditions that most of the correct ingredients satisfy, while incorrect ones do not. Using the condition set, we show that more than 93% of ingredient candidates can be screened out while achieving 99% of true positive rate.

2. Methodology

We examine more than 50K one-line replacement patches to examine whether the correct ingredients share some common properties. Then, based on the properties, we establish a set of conditions that most of the correct ingredients satisfy.

2.1 Dataset

Table 1. the statistics of the CodRep corpus

Dataset	# data	LoC
1	3,858	2,056,900
2	10,088	5,388,282
3	15,326	627,593
4	10,431	2,308,279
5	18,366	2,785,599

The CodRep corpus [9] contains unique 58,069 one-line replacement patches collected from open-source Java projects. Table 1 shows the statistics of the data. Each data is a triplet (I, F, n), where I is a correct (ground-truth) ingredient, F is a file containing the faulty line (a list of source code lines), and n is the line number of the faulty code in F (line numbers starts from 1). Figure 1 is an example of a data point.

I	(value="org.apache.openjpa.persistence.meta.AnnotationProcessor6",date="Fri May 14 16:05:41 EDT 2010") (A)
F	/**↲ * Generated by OpenJPA MetaModel Generator Tool.↲ **/↲ ↲ package openbook.domain;↲ ↲ import javax.persistence.metamodel.SingularAttribute;↲ ↲ @javax.persistence.metamodel.StaticMetamodel↲ (value=openbook.domain.Customer.class)↲ @javax.annotation.Generated↲ (value="org.apache.openjpa.persistence.meta.AnnotationProcessor6",date="Thu May 13 20:18:23 CDT 2010")↲(B) public class Customer_ {↲ public static volatile SingularAttribute<Customer,String> email;↲ public static volatile SingularAttribute<Customer,Long> id;↲ public static volatile SingularAttribute<Customer,String> name;↲ public static volatile SingularAttribute<Customer,Integer> version;↲
n	**12**

Figure 1. the example of CodRep data

2.2 Properties of Correct Ingredients

For all data points in the CodRep corpus, we measure the relationship between I (repair ingredient) and F[n] (faulty line) using 7 metrics[1] described in Table 2. For example, in the case of Figure 1, we compared (A) and (B). The values of the last three metrics are subsequently normalized in the data point level as follows:

$$for\ cm\ \in \{LCS, NMB, MBS\},$$

$$cm(I, F[n]) = \frac{cm(I, F[n])}{\max_{line\ \in\ F_{ne}\ \cup\ \{I\}} cm(line, F[n])},$$

where F_{ne} is a set of non-empty lines in F. Hereafter, we will refer the set $F_{ne} \cup \{I\}$ as the *ingredients pool*.

Table 2. the comparison metrics for strings A and B

metric	description
BoW [10]	The Bag of Words similarity (term frequencies)
bBoW	The Bag of Words similarity (binary)
LS	The strings' similarity as a float in the range [0, 1]. When T is the number of total characters in A and B, and M is the number of matches, LS(A, B) = 2.0 * M / T.
ED [11]	Edit Distance, i.e., the number of operations needed to transform A into B
LCS	The length of the longest common sequence
NMB	The number of matching blocks (non-overlapping matching subsequences)
MBS	The total size of the matching blocks

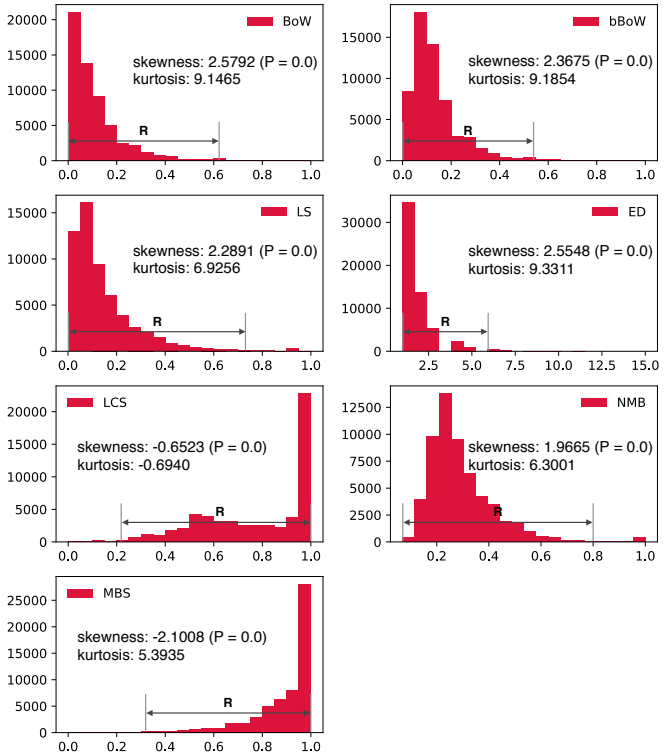


Figure 2. the histograms for each metric (bins=20)

Figure 2 contains histograms that show the distribution of the values of each metric. We note that all histogram is skewed left or right [12]: for example, the histogram of LS has a long right tail, which means that LS

[1] The implementation is available at https://github.com/agb94/KCSE19.

has a right-skewed (positive-skew) distribution, the mean of which is larger than the median. Based on this skewness information, we will establish a set of conditions that most of the correct ingredients satisfy in Section 2.3.

2.3 Properties to Conditions

Once we build a model that describes or captures the properties of the correct ingredients, the model may be used to screen any unseen ingredient candidates. So, we defined a ***plausible range R*** for each metric ***cm*** as follows:

- When *cm* has a right-skewed distribution,
  R = the range of leftmost t% values
- otherwise,
  R = the range of rightmost t% values.

If *t* is 100, R is identical to the original range of the metric. The smaller the t value is, the narrower the ranges becomes. A smaller t eventually means a more rigorous screening procedure, since the ranges will be used to filter the ingredient candidates. Here we set *t* to 99 to cover the majority of values coming from correct ingredients. As presented in Figure 2, the R values obtained from the CodRep dataset are as follows:

- BoW : [0.0, 0.6220]
- bBoW : [0.0, 0.5412]
- LS : [0.0005, 0.7333]
- ED : [1, 6]
- LCS : [0.2188, 1.0]
- NMB : [0.0690, 0.8]
- MBS : [0.32, 1.0]

This set of ranges forms a condition set which will be used during the screening procedure described in the next section.

2.4 Screening Procedure

In practice, when we apply a replacement operator on the given faulty line, we may have several ingredient candidates that form the original ingredient search space. For example, at the file-scope, the ingredient pool will be the set of unique lines in the file containing the faulty line. For each candidate, we first calculate the value of each metrics in Table 2 and then check whether each value is within the corresponding range in the condition set presented in Section 2.3. The score of a candidate is defined as the number of satisfied conditions; the maximum score is the size of the condition set (here, 7), and the minimum is zero. At the final step, the candidates whose scores are less than ***s*** is screened out so that the ingredient search space is reduced.

3. Evaluation

In the previous section, we established a condition set for a repair ingredient based on the statistical investigation of real-world Java commits using various metrics. And then we suggested the screening procedure for ingredient candidates given a faulty code. In this section, we will evaluate the proposed method in terms of both effectiveness and efficiency.

3.1 Research Questions

- RQ 1. (*Effectiveness*) How many conditions are met for the correct ingredients in general? Do the incorrect ingredients satisfy fewer conditions than the correct ingredient?
- RQ 2. (*Screening Performance*) How much of the ingredient search space can be reduced using the screening procedure?
- RQ 3. (*Efficiency*) How long does it take to reduce a search space?

3.2 Environment

The experiment has been conducted on a PC with Intel Core i7-7700 CPU and 32GB memory running Ubuntu 16.04. We use *sklearn* version 0.19.1 and Python 3.6.2's standard library *difflib* to compute the lexical features described in Table 2.

3.3 Result

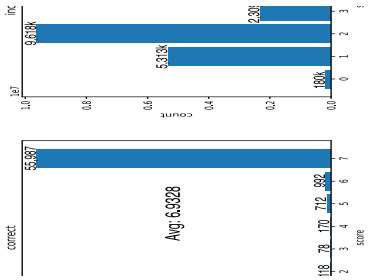


Figure 3. the score frequency distributions for the correct and incorrect ingredients in the CodRep corpus (t = 99)

3.3.1 Effectiveness (RQ 1). An ingredient search space is defined as the set of unique lines in the file containing given faulty line. In the space, there may be numerous candidates which are not identical to the ground-truth *correct* ingredient (I) or the faulty line (F[n]). Considering such candidates as the *incorrect* ingredients, we compare the score distributions of the correct and incorrect ingredients (Figure 3). The average score of the incorrect ingredients is only 2.1938, whereas that of the correct ingredients is almost 7. The incorrect ingredients satisfy much fewer conditions than the correct ingredients.

Table 3. the reduction ratio of ingredient search space (st = score threshold)

st	**0**	**1**	**2**	**3**
reduction ratio	0.0	0.009	0.276	0.76
st	**4**	**5**	**6**	7
reduction ratio	0.876	0.934	0.966	0.984

3.3.2 *Screening Performance (RQ2).* Suppose we reduce the ingredient search space to the set of ingredients whose scores are equal to or larger than ***st***, score threshold. The reduction ratio is then defined by the ratio between the number of filtered out ingredients and the number of total ingredients. Table 3 presents the

reduction ratios corresponding to each threshold. We observe that the reduction ratio is greater than 93% with the score threshold of 5 and over, which means that we can discard more than 90% of ingredients without running the target program. Also, from the RQ1, we confirm that 99.35% of correct ingredients has a score of 5 and over, which means the reduced search space will contain the correct ingredient with a high probability. To sum up, if we filter out the ingredients whose score is below than 5, we can expect more than 90% of search space reduction while achieving about 99% of true positive rate.

3.3.3 Efficiency (RQ3). In the CodRep corpus, the average size of the initial search space is 342.42. We have applied the screening procedure to all data points and measured how long it takes to compute the scores of entire candidates. As a result, it takes about 80 minutes on average to process the entire 58,069 data points, which means only 0.083 seconds are required to process one data point. Therefore, it would not yield large overhead to place the screening procedure between the fault localization and the patch generation in the APR process.

4. Discussion

In this paper, we only consider one-line replacement patches. Since there are unlimited types of fixes, one can regard this type of patches as just little subset of real-world bug fixing. However, we have found that about 20% of faults in Defects4J [13], which is a database of real faults, can be patched by one-line replacement patches; this suggests that a non-negligible portion of real-world faults may be fixed by the simple operator/patch. Currently, this one-line or -statement replacement operators are employed by many G&V repair techniques, and those operators consume large amounts of resources due to the cost of exploring huge search spaces. Therefore, we expect that using our screening technology will significantly reduce the cost by effectively curtailing the search space.

We have validated the condition set only on the CodRep corpus, which is also used to build the set. As it contains more than 50 thousands real-world patches, we expect that our screening procedure is sufficiently generalizable, at least for Java. However, as future work, we plan to evaluate this procedure using other benchmarks that are unseen during the construction of the conditions.

5. Conclusion

We present a novel ingredient screening procedure for one-line replacement operator, which is a typical edit operator employed by G&V techniques. Based on the CodRep corpus that consists 58,069 one-line replacement patches, we build a condition set that almost all correct ingredients satisfy. We introduce a screening procedure using the condition set and show that the procedure filter out more than 93% of incorrect ingredients while maintaining 99% or true positive rate. Also, we find that the reduction procedure only takes about 0.1 second per datapoint, which is a genuinely negligible cost, for file-scope ingredients search space.

Acknowledgement

This work is supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (No. 2017M3C4A7068179) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (Grant No. NRF-2016R1C1B1011042).

References

[1] Rinard, M.C., Technical perspective patching program errors. Communications of the ACM, 51(12), pp. 86-86. (2008)

[2] Harman, M., Automated patching techniques: the fix is in: technical perspective. Communications of the ACM, 53(5), pp. 108-108. (2010)

[3] Weimer, W., Nguyen, T., Le Goues, C. and Forrest, S., Automatically finding patches using genetic programming. In Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society. pp. 364-374. (2009)

[4] Xuan, J., Martinez, M., Demarco, F., Clement, M., Marcote, S.L., Durieux, T., Le Berre, D. and Monperrus, M., Nopol: Automatic repair of conditional statement bugs in java programs. IEEE Transactions on Software Engineering, 43(1), pp. 34-55. (2017)

[5] Barr, E.T., Brun, Y., Devanbu, P., Harman, M. and Sarro, F., The plastic surgery hypothesis. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM. pp. 306-317. (2014)

[6] Martinez, M., Weimer, W. and Monperrus, M., Do the fix ingredients already exist? an empirical inquiry into the redundancy assumptions of program repair approaches. In Companion Proceedings of the 36th International Conference on Software Engineering. ACM. pp. 492-495. (2014)

[7] Xin, Q. and Reiss, S.P., Leveraging syntax-related code for automated program repair. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering. IEEE Press. pp. 660-670. (2017)

[8] Wen, M., Chen, J., Wu, R., Hao, D. and Cheung, S.C., Context-Aware Patch Generation for Better Automated Program Repair. ICSE. (2018)

[9] Chen, Z. and Monperrus, M., The CodRep Machine Learning on Source Code Competition. arXiv preprint arXiv:1807.03200. (2018)

[10] Harris, Z., Distributional structure. (J. Katz, Ed.) Word Journal Of The International Linguistic Association, 10 (23), pp. 146-162. (1954)

[11] Navarro, G., A guided tour to approximate string matching. ACM computing surveys (CSUR), 33(1), pp. 31-88. (2001)

[12] Kokoska, S. and Zwillinger, D., CRC standard probability and statistics tables and formulae. Boca Raton, Fla.: Chapman & Hall/CRC. (2000)

[13] Just, R., Jalali, D. and Ernst, M.D., Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In Proceedings of the 2014

International Symposium on Software Testing and Analysis. ACM. pp. 437-440. (2014)

소프트웨어 제품 라인 개발을 위한 가변성 모델의 무모순성 검증

한영훈, 강성원, 이동민, 박병훈 ○

한국과학기술원, 티쓰리큐 ○

{younghun.han, sungwon.kang, theresaldm}@kaist.ac.kr, warmpark@t3q.co.kr ○

Verifying Consistency of Variability Models for Software Product Lines

Younghun Han, Sungwon Kang, Dongmin Lee, Byung-Hoon Park ○

KAIST, T3Q○

요 약

우리는 과거 연구에서 기존 소프트웨어 제품 라인 가변성 모델링 방법의 여러 문제점을 해결한 새로운 가변성 모델링 방법인 CFM(Compositional Feature Modeling)에 기반한 가변성 모델링 도구 SVM(Systematic Variability Modeler)를 개발하였다. SVM 분석기는 SVM이 구축한 가변성 모델을 분석하기 위한 도구로, 궁극적으로 효율성 측면의 기존 도구의 한계점을 개선하기 위한 도구이나, 우선 이 도구가 가변성 모델이 올바른 지를 판단할 수 있어야 한다. 본 논문에서는 SVM을 위해 정확성 측면에서 가장 중요한 관점인 '무모순성(consistency) 분석'의 방법을 개발하여 검증한다.

1. 서론

소프트웨어 제품 라인 개발에서 제품들 간의 공통성과 가변성을 분석하는 1)가변성 모델링과 2)가변성 모델 분석은 소프트웨어 제품 라인 개발에서 가장 핵심적인 작업이다. 체계적인 가변성 모델 분석을 수행하기 위해서는, 가변성 모델링 방법 연구가 선행되어야 한다. 우리는 [1-3]의 연구에서 기존 가변성 모델링 방법의 여러 문제점을 해결한 새로운 가변성 모델링 방법 CFM과 도구 SVM을 개발하였다.

가변성 모델 분석과 관련 된 연구들은 2000년 이후부터 활발하게 연구 되고 있으며, 가장 대표적인 연구들은 Solver 기반의 분석 방법론이다[4]. 하지만, 이 연구들은 정확성(correctness) 및 효율성(efficiency) 측면에서 몇 가지 한계점이 있다. 우리는 이러한 한계점들을 개선한 SVM 분석기를 개발 하고 있다. SVM 분석기는 궁극적으로 효율성 측면의 기존 도구의 한계점을 개선하기 위한 도구이나, 우선 이 도구가 가변성 모델이 올바른 지를 판단할 수 있어야 한다. 본 논문에서는 정확성 측면에서 가장 중요한 관점인 '무모순성(consistency) 분석'의 방법을 개발하여 검증한다.

2. 기존 가변성 모델 분석 방법의 한계점

기존 가변성 모델 분석 방법은 정확성 측면과 효율성 측면에서 몇 가지 한계점이 있다. 첫째, 정확성 측면에서 가변성 모델 분석 방법들이 항상 올바른 답을 제시하는지 검증 되지 않았다. 둘째, 가변성 모델 분석의 효율성 측면에서 1)가변성 모델을 구성하는 피처의 수가 증가함에 따라 분석 규모가 지수적(exponential)으로 증가 하는 한계점이 있다. 피처의 수가 100개, 200개 및 300개 일 때, 가변성 모델 분석의 최악의 시간복잡도는 $O(2^{100})$, $O(2^{200})$, 및 $O(2^{300})$이다. 2)가변성 모델이 Solver의 입력이 가능한 형태로 변환되면서, 피처들 간의 트리 구조 및 제약사항 구조와 같은 아키텍처적인 요소들을 잃어버리기 때문에 분석 결과에 대한 원인을 추적하는 것이 매우 어렵다는 한계점이 있다. 마지막으로, 3)분석 관점 간의 관련성이 있음에도 불구하고, 관련성 분석 기능이 없는 단일한 Solver로 다양한 분석 관점을 동시에 분석하기 어려운 한계점이 있다. 가변성 모델링 방법 및 분석 도구가 현실적인 문제에 적용 할 수 있는 수준이 되려면, 위에 제시한 정확성 및 효율성 측면의 한계점들이 개선되어 실용적인(practical) 수준이 되어야 한다.

이러한 한계점들 중에서도 우리는 정확성 측면에 해당되는 관점인 '무모순성 분석' 방법을 SVM 분석기를 위해 개발하고 그 분석방법의 정확성을 검증하였다.

3. SVM 분석기: 가변성 모델 분석 도구

우리가 개발한 가변성 모델 분석 도구 SVM 분석기의 분석 절차는 5단계로 구성 된다.

1단계에서는 가변성 모델을 도구가 분석할 수 있는 형태로 변환한다. 즉, 1단계 산출물로 'Feature Matrix'가

생성 된다. ‘Feature Matrix’는 가변성 모델이 가지는 아키텍처적인 정보들도 가지고 있으며, 가변성 모델과의 추적성 관계도 지원하기 때문에 분석 결과에 대한 원인 식별에도 활용 될 수 있다.

2단계에서는 가변성 모델에서 제약사항들을 식별하고, 특성을 분석한다. 가변성 모델은 피처 트리로만 구성 된 모델도 있고, 피처 트리에 제약사항까지 함께 구성 된 모델도 있다. 피처 트리로만 구성 된 모델에는 특정 분석 관점에 대한 문제점이 발생하지 않는 다는 것이 과거연구에서 증명되었기 때문에, 많은 연구자들이 피처 트리와 제약사항간의 의미론적 충돌 때문에 가변성 모델에서 다양한 문제점들이 발생한다는 의견들이 지배적이다. 2단계에서 제약사항을 식별하고 특성을 분석하는 이유는 한 제약사항이 다른 제약사항에 영향을 주지 않고 독립적으로 존재할 수도 있고, 영향을 줄 수도 있기 때문에 이 문제를 효율성 측면에서 개선하려면 서로 분리해서 분석하여야 한다.

3단계에서는 독립적인 제약사항들을 분석 관점에 따라 분석을 한다. 우리는 독립적인 제약사항들과 피처 트리 간 발생할 수 있는 모든 경우의 수를 사전에 분석하여, 그 결과를 SVM 분석기가 활용할 수 있도록 저장소에 보관하고 있다.

4단계에서는 다른 제약사항에도 영향을 주는 제약사항들 간의 분석을 수행한다. 이 단계에서는 2개 이상의 제약사항을 분석하는 과정이다. 이 과정에서는 2, 3, 4개 이상의 연관성 있는 제약사항들을 분석하려면, 지수적으로 분석의 범위가 늘어난다. 우리는 이러한 문제점을 해결하기 위해서, ‘Feature Matrix’안의 가변성 모델 아키텍처적인 요소들을 활용하여, 비교적 쉽게 분석의 범위를 줄 일 수 있었다.

5단계에서는 3-4단계 분석 결과를 종합하여 분석 관점 단위로 최종 분석 결과를 제시하고, 이에 대한 원인까지 추적해 준다.

4. 평가

우리는 [5-6]의 연구에서 제공하는 시험 항목 활용하여, 우리가 개발 한 도구의 정확성을 시험한다. 논문 [6]은 총 7개의 분석 관점에 대해서 총 192개의 시험 항목을 제공한다. 본 논문의 평가에서는 다양한 분석 관점들 중에서 가장 중요한 분석 관점인 ‘무모순성 분석’ 실험만 진행한다. 이 분석 관점을 실험하기 위해서 제공되는 시험 항목의 수는 총 24가지이다. 우리는 [6]의 24개 시험 항목을 활용하기 위해서, 시험 항목의 입력을 가변성 모델링 도구 SVM을 활용하여 모델링 하였다. 그 후, 24개 시험 항목에 대해서 [6]에서 제시한 출력과 SVM 분석기 결과와 비교한다. 각 시험 항목에 대해서 10회, 총 240회 반복하여 실험하였다. 총 24개 시험 항목의 대해서 [6]에서 제시한 결과와 SVM 분석기의 결과가 100% 일치하였다.

5. 결론

우리는 기존 가변성 모델 분석 연구들의 한계점을 정확성 및 효율성 측면에서 제시하였다. 이러한 한계점을 개선하기 위해 SVM 분석기를 개발 하고 있다. 가변성 모델 분석 도구의 효율성을 개선하기 위해서는 어떠한 경우에서도 올바른 답을 제시하는 정확성 측면의 검증이 선행되어야 하므로, 본 논문에서는 ‘무모순성 분석’의 측면을 검증하였다. 시험 항목을 활용 한 검증 결과 100% 합격의 결과를 얻었지만, 이것이 본 도구가 가변성 모델의 ‘무모순성 분석’ 관점에서 완전히 정확하다는 것을 보장하는 것은 아니다. 우리는 향후 정확성 측면의 검증을 보다 엄격하게 수행하기 위해서 수학적인 증명을 할 계획이며, 그 외의 다른 분석 측면들을 위한 방법들도 만들어나갈 계획이다.

Acknowledgement

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업 의 지원을 받아 수행된 연구임. (NRF-2017M3C4A70 66210)

참고문헌

[1] Kang, S., Han, Y., Ahn, H., Lee, J., “Critical analysis of feature model evolution,” The 8th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, November 24-26, 2017.

[2] 한영훈, 강성원, 박연하, 박병훈, "SVM: 소프트웨어 제품 라인 개발을 위한 가변성 모델링 도구," 제20회 한국 소프트웨어공학 학술대회(KCSE), 2018.

[3] 강성원, 한영훈, 안휘, 이지현, "소프트웨어 제품 라인 개발을 위한 합성적 피처 모델 CFM과 CFM을 지원하는 가변성 모델링 도구 SVM," 정보과학회지, 36(4), 9-18, 2018.

[4] D. Benavides, S. Segura, and A. Ruiz-Cort´es, “Automated analysis of feature models 20 years later: a literature review,” Information Systems, 35(6), 2010.

[5] S. Segura, D. Benavides, and A. Ruiz-Cortés, "Functional testing of feature model analysis tools: a test suite." IET software 5.1 (2011): 70-82.

[6] S. Segura, D. Benavides, and A. Ruiz-Cort´es, “FaMa Test Suite v1.2,” Technical Report ISA-10-TR-01, ISA research group, 2010.

Automatic Data Flow Analysis to Generate Test Cases from Activity Diagrams

Aman Jaffari, Cheol-Jung Yoo, and Jihyun Lee
Dept. of Software Engineering, Chonbuk National University
{aman, cjyoo, jihyun30}@jbnu.ac.kr

Abstract

Recently, testers adopting model-based testing has shifted their focus on utilizing the behavioral models of the system under test as test basis for automated test generation. Still, the existing studies focus mainly on detecting various control flow errors. Thus, they are missing the data flow aspects. Moreover, conventional data flow testing methods are mostly based on the code's flow graph. This paper introduces an approach to automatically analysis data flow from activity diagrams. An experimental investigation is performed to illustrate the effectiveness of the proposed approach, using the faulty version of a renowned problem. The results show that the proposed approach can generate relatively effective test cases for detecting faults that occur by improper use of data in computation and decision while benefiting from the advantages of model-based testing.

1. Introduction

The acceptance of UML diagrams as the de-facto standard for modeling software systems expanded the interest in model-based testing (MBT). MBT is a technique for identifying a set of test cases from a model representing the expected behavior of the system under test (SUT) [1]. More recently, testers adopting MBT technique has shifted their focus on utilizing the behavioral models of the SUT [2]. Accordingly, having a rich symbolic vocabulary, the activity diagram (AD) is regarded as the most powerful and comprehensive design artifact for modeling the intended behaviors of the SUT [3]. The existing studies have investigated on the AD-based testing mostly from the high-level business process (HLBP) for the purpose of test automation, saving time and effort while detecting various control flow errors, e.g. loops, and synchronization [4-7].

Most programs deliver functionality in terms of data consequently errors easily can occur if the flow of data is inappropriately modeled, defined, and used. The underlying idea behind data flow testing (DFT) is detecting inappropriate use of data. This idea well suites the AD-based approach, since the AD represents the sequence of actions among activities that embody lower-level steps in the overall activity and are central to the data flow aspect of activities [8].

The objective of this paper is to utilize the AD of the SUT for automatically tracking the flow of data between activities. For this purpose, we have provided an algorithm for the automatic identification of data flow information (DFI). To investigate the applicability of the proposed technique, we have performed an experimental investigation using the faulty version of the commission problem that contains a mix of computation and decision making with interesting data flow properties. The results with statistical evidence indicate that the proposed technique is relatively as effective as the conventionally well-known testing technique while benefitting from the biggest advantages of MBT like mastering complexity, test automation, reuse of models, and test efficiency.

Section 2 presents the background and related work. Section 3 introduce the main approach. Section 4 discusses an experimental study that was performed to evaluate the proposed approach. Finally, conclusion and future work are presented in Section 5.

2. Background and Related Work

Currently, there are many studies in the literature with different strategies using the AD for automatic test generation. The AD models are considered useful for testing due to having a very powerful symbolic vocabulary for expressing the system's behavior and the ability to be manipulated by automated means [1]. An AD-based approach for the synthesis of test scenarios introduced by [7]. Also, [4] present a so-called enhanced test case generation technique based on the AD. Another study by [5] uses condition-classification tree method to generate test cases from the AD. A novel approach based on AD with use case scope is introduced by [6]. A gray-box approach by [9] generates test cases directly from the AD with the purpose of combing white box and the black box methods. In our previous work, we have manually annotated the AD with DFI to facilitate DFT for IoT mobile applications [10].

DFT is introduced to supplement the limitation of control flow testing (CFT). Many empirical studies in the literature have claimed its effectiveness over the CFT. Yet, conventional DFT methods are strictly based on the static analysis of source code [11]. Some studies also have applied DFT technique to behavioral models. For example, a Petri nets-based approach by [12] construct a reachability graph with data operation and then reduce the reachability graph to detect data flow errors. Also, [13] apply DFT techniques to abstract state machines.

Currently, AD-based methods are focused on test automation mainly from the HLBP analyzing different control flow information which is missing the important aspects (DF) and capabilities of ADs. Likewise, DFT methods are mostly based on the static analysis of source code which misses the dynamic part of DFI [11]. We believe examining the ADs from a lower-level can allow automatically analyzing and identifying DFI. To the best of our knowledge, automatic DFT has never been considered

based on ADs. Therefore, this study examines the automatic identification of definition-use pairs (du-pairs) using AD.

3. The Approach

This section introduces the main concepts for automatic DFT based on AD, providing an example showing how the data flows among activities, the basic definitions, and an algorithm.

3.1 Data flow representation and concept in AD

An activity is a behavior specified as sequencing of subordinate units, using a control and data flow model [8]. Actions are executable nodes required for any significant capabilities of activities that embody lower-level steps in the overall activity. Action notation only appears in AD and are central to the DF aspects of activities [8]. Actions can invoke other behaviors and operations, access and modify objects, and perform more advanced coordination of other actions, for example, Figure 1(a) top left activity has two actions: a CreateObject action that creates an object of type Activity B, and a CallBehavior action that calls Activity B. Actions also have a specialized form of object node (Pins) for object flows to get and provide data. For example, Figure 1(b) bottom left activity has a SendObject action that sends some object via its output Pin and another action that gets the object via its input Pin and manipulates its contents.

These object flows provide primary means for exchanging data between activities. Activities also introduce parameters and structural features or variables. As depicted in Figure 1(c) right, the use of variables in AD effectively provides indirect data flow paths from the point at which a value is written to the variable to all the points at which the value is read from the variable [8]. By analyzing the action sequence and their behavior, and utilizing the information appeared in action Pins, activity parameters, decision constraints, it possible to automatically track the flow of data among activities in an AD.

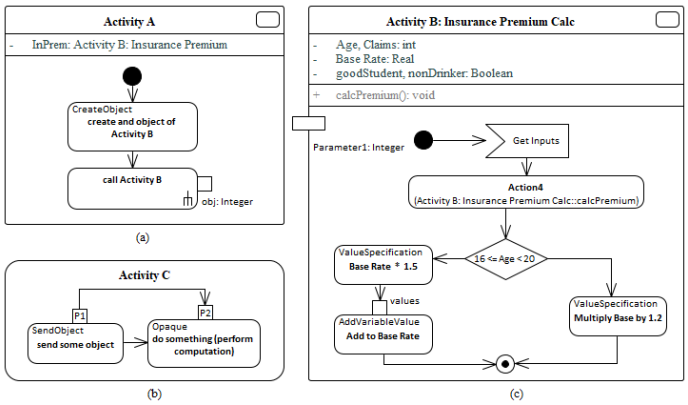


Figure 1. An illustrative example showing data flow concept in AD

3.2 Definitions and Algorithm

Definition 1. An AD is a directed graph that can be described formally as an activity model, $M = (A, E, C, V, a_I, a_F)$ where:

- $A = \{a_1, a_2, ..., a_n\}$ is a set of executable nodes (actions are the only kind of executable node and object nodes are attached to actions as action Pin, by executable nodes, we indicate the lower-level steps in the overall activities)
- E denotes a set E of edges, where $E \subseteq \{A \times A\}$
- $C = \{c_1, c_2, ..., c_n\} = D_N \cup J_N \cup F_N \cup M_N$ is a set of control nodes such that: D_N is a set of decision nodes; J_N is a set of join nodes; F_N is a set of fork nodes, and M_N is a set of merge nodes. In terms of data flow, D_N is analogous with p-use that stands for predicate use.
- a_I is an initial activity node, where $a_I \subseteq A$ and $a_I \neq \emptyset$
- a_F is a set of final nodes, where $a_F \subseteq A$ and $a_F \neq \emptyset$.
- $V = \{v_1, v_2, ..., v_i\}$ is a set of variables s.t. V includes both object variables and structural attributes (variables).

Definition 2. Variables can appear in different contexts in AD, let M be a model that has a set of variable V:

- We say variable $v_i \in V$ is *defined* in node a_n of M, indicated as $def_{a_n}^{v_i}$ meaning that variable v_i is defined in node a_n, if v_i is initialized in the corresponding node or its value has been modified by the execution of certain action.
- We say variable $v_i \in V$ is in predicate use in decision node d_n of M indicated as $pu_{d_n}^{v_i}$ meaning that variable v_i is in *pu* in decision node d_n if it appears as constraint of D_N.
- We say variable $v_i \in V$ is in computation use in node a_n of M indicated as $cu_{a_n}^{v_i}$ meaning that variable v_i is in *cu* in node an if the corresponding node behavior is involved with any sort of computation.

Algorithm 1 describes the generation of du-pair paths from AD. The algorithm takes a model M of AD in the form of XML, a given node c which is the current node, an initial node a_I, a variable v, and a path P. Starting from the given node c for c in M and c is not in P, the algorithm check its properties for the corresponding v if the v is not in *def* append c to P, set the incoming node as current node, make recursive call, the iteration stops until it finds the node that v is in *def* and return the path P. Simply the algorithm find the *def* of v_i used in a given node anywhere in the model searching backward.

Algorithm 1. *ExtractDuPairs (M, c, a_I, v, P[])*: The algorithm for automatically generating the du-pair paths.

Input: An activity diagram model M, a current node c, an initial node a_I, a variable v and a list of path P:
Outputs: P: A list of du-pair paths

```
1: Begin
2: P = P + [c]    # add current node to P
3:   if c == a_I   return P
4:   for node c incoming edges, e = (n, m) s.t n ∈ M, e ∈ E do
5:     if c in M and node c has not been visited then
6:       record edge e as the discovery edge for node c
             #get previous node p
7:       for v_i in c:    #Check the occurrence of v in c
8:         check the type of node c: #depending on the kind of node
9:           append c to P as def_{a_n}^{v_i} or pu_{a_n}^{v_i} or cu_{a_n}^{v_i}
10:      endfor
11:      s = p    #set the previous node as start node
12:      Recursively call ExtractDuPairs(M, s, a_I, v, P)
13:    endif
14:  endfor
15:  return P
16: End
```

4. The Experiment

The purpose of this experiment is to investigate the applicability of the proposed technique and evaluate its effectiveness against the conventional testing. Therefore, we formulate the following research question RQ: Is it possible to track the flow of data among activities in an AD to automatically generate test cases? Also, we define the corresponding hypothesis. Null hypothesis (H0): there is no significant difference between the two approaches in terms effectiveness. Alternative hypothesis (Ha): There is a significant difference between the two approaches in which the conventional technique is outperforming over the proposed technique. Because the purpose of the experiment is to test the applicability of the proposed approach, hence we are initially assuming from the perspective that the conventional approach is more effective. Accordingly, if the H0 is not rejected the answer to our RQ is positive, determining its applicability.

The experimental investigation is performed on the faulty version of a problem known as the commission problem. The commission problem is an interesting problem for DFT, due to having a mix of computation and decision making. Paul [14] has performed different experiments with different testing techniques using this problem. All of his experimental results are available on the CRC website. Among them we have chosen the experimental result reported based on the equivalence class testing. The perfectness and redundancy reductions are the two major motivation of equivalence class testing. There are various techniques of equivalence classes from traditional to improved, weak and strong robust equivalence testing [14]. In our comparative analysis, the strong robust equivalence testing is preferred that seems to be relatively more effective.

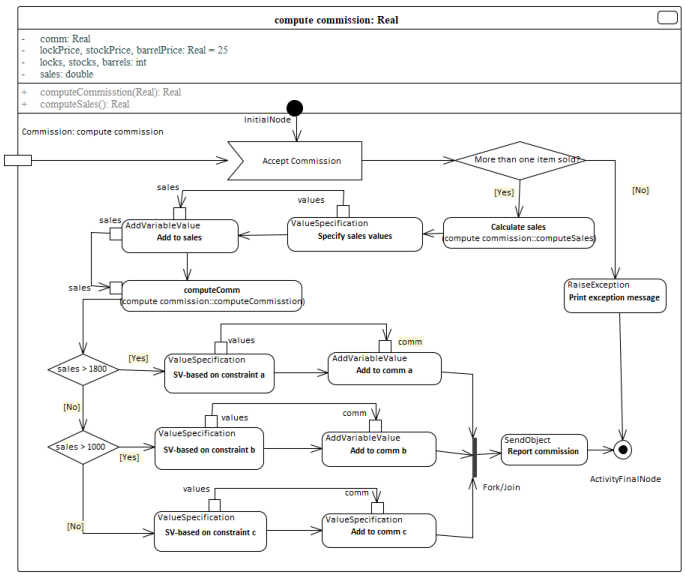


Figure 2. Activity diagram for the commission problem

Figure 2 depicts the AD for the commission problem. The AD is first exported as an XML file, and the du-pairs are automatically identified. The *du* of variables in the AD is determined either based on the encapsulated behavior of actions, simply its kind or by decision constraints. The decisions are always analogous with p-use. The commission AD has a base-action which accept the commission object from other activity. We assume that the base-action must initialize all the variables that are involved within the activity. Hence, it is an action that variables are defined in. There are three decision nodes, all variables appearing in the decision nodes are regarded as p-used. The remaining actions in the AD are analyzed in terms of their kinds*, for instance, the ValueSpecification action specifies the value for variable sales using the price and numbers of items sold (e.g. locks, stocks, and barrels) and pass the values to the next action via its output Pin. The AddVariableValue action gets the values via its input pin and redefines the sales variable. Similarly, the rest of the actions are analyzed.

Table 1 presents the corresponding generated testing paths for Figure 2. For brevity and readability purposes, we have presented the identified testing paths in the form of a table which lists the variable objects corresponding to their def-use information. Based on the selected family of data flow coverage criteria we can apply different coverage criteria and practice a different combination of the identified paths. In this experiment, we have adapted the family coverage criteria known as definition-clear path coverage criteria introduced by [14]. Also, we have not covered all variables in an AD but a subset of critical variables.

Table 1. The generated du-pair testing paths

variable object	def	p-use	c-use
locks	Accept Commission	More than one item sold?	Specify sales value
stocks	Accept Commission	More than one item sold?	Specify sales value
barrels	Accept Commission	More than one item sold?	Specify sales value
sales	Add to sales	sales > 1800	SV-based on constraint a
sales	Add to sales	sales > 1800 sales > 1000	SV-based on constraint b
sales	Add to sales	sales > 1800 sales > 1000	SV-based on constraint c
comm	Add to comm a	None	Report commission
comm	Add to comm b	None	Report commission
comm	Add to comm c	None	Report commission

4.1 The results and discussion

This section presents the experimental results and discusses the outcomes in terms of the test effectiveness. Table 2 presents a brief summary of the reported results obtained by performing test with the two techniques namely AD-based and robust equivalence class testing. To answer the formulated RQ and test the associated hypothesis a simple statistical analysis is performed on the summarized data. We have performed a two-sample proportion test with a p-value approach to decide if there is statistically sufficient evidence at the $\alpha = 0.05$ level to show the differences between the two techniques.

Table 2 A brief summary of the reported results

Techniques	*Total*	*fail*	*pass*	*p-value*
AD-based (Du-pairs)	21	16	5	0.4399
Robust Equivalence	27	23	4	

* Due to space limitation interested readers on the action kind and behavior can refer to the OMG UML documentation [8]

"OMG® Unified Modeling Language® (OMG UML®) Version 2.5.1," OMG, 2017.

As a result, the H_0 is accepted indicating no difference between the two testing techniques. Figure 3, presents a simple comparison of the two technique's sample proportion and their differences in terms of the number of failed test cases. For the purpose of this paper, we have focused on the validation of the AD-based testing with data flow information in contrast to the conventional testing, to assure if we are able to analyze data flow information among activities automatically. Meanwhile, the results indicated a good performance of the AD-based testing in terms of effectiveness. Reaching a certain level of effectiveness that is comparable with a well-practiced technique is quite remarkable. Moreover, the AD-based approach showed better efficiency regarding the real fault detected and the number of tests executed.

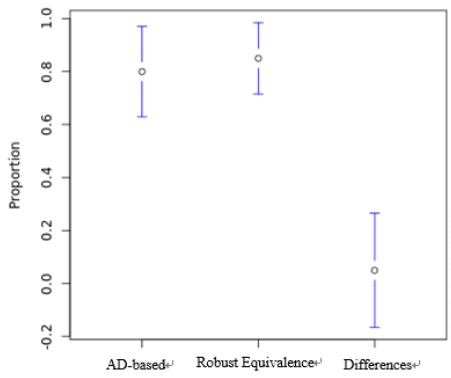


Figure 3. Proportion comparison of the two techniques

In summary, the answer to our RQ is quite positive whereas the experimental results sufficiently validated its applicability in contrast to a well-practiced testing technique. Based on the lesson learned through the experiment, we can conclude that in addition to control flow errors AD-based approach with data flow information can detect faults that occur by improper use of data in computation and decision. Furthermore, the approach can completely benefit from all the advantages of model-based testing such as mastering complexity, test automation, reuse of models, test efficiency, and etc.

5. Conclusion and future work

This paper presents a technique to automatically analyze data flow information among activities in an AD. Experimental data show that AD-based approach has the ability to detect faults that occur by improper use of data in addition to control flow errors while benefiting from the advantages of MBT. We have presented an algorithm to support automatic AD-based data flow analysis. In the future, we intend to extend our approach for further improvement. We also want to evaluate our approach in different cases with different size and complexity.

Acknowledgment. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A3B03028609).

References

[1] P. Jorgensen, *The craft of model-based testing*, Boca Raton: CRC Press/Taylor & Francis Group, 2017.

[2] M. Shirole, and R. Kumar, "UML behavioral model based test case generation: a survey," *SIGSOFT Softw. Eng. Notes,* vol. 38, no. 4, pp. 1-13, 2013.

[3] M. Felderer, and A. Herrmann, "Comprehensibility of system models during test design: a controlled experiment comparing UML activity diagrams and state machines," *Software Quality Journal*, April 23, 2018.

[4] P. N. Boghdady, N. L. Badr, M. A. Hashim, and M. F. Tolba, "An enhanced test case generation technique based on activity diagrams." The International Conference on Computer Engineering & Systems, pp. 289-294, 2011.

[5] S. Kansomkeat, P. Thiket, and J. Offutt, "Generating test cases from UML activity diagrams using the Condition-Classification Tree Method." ICSTE, pp. 289-294, 2010.

[6] D. Kundu, and D. Samanta, "A novel approach to generate test cases from UML activity diagrams," *Journal of Object Technology,* vol. 8, pp. 65-83, 2009.

[7] A. Nayak, and D. Samanta, "Synthesis of test scenarios using UML activity diagrams," *Software & Systems Modeling,* vol. 10, no. 1, pp. 63-89, 2011/02/01, 2011.

[8] "OMG® Unified Modeling Language® (OMG UML®) Version 2.5.1," OMG, 2017.

[9] L. Wang, J. Yuan, X. Yu, J. Hu, X. Li, and G. Zheng, "Generating test cases from UML activity diagram based on Gray-box method." 11th Asia-Pacific Software Engineering Conference, pp. 284-291, 2004.

[10] A. Jaffari, J. Lee, C.-J. Yoo, and J. H. Jo, "Test Case Generation Technique for IoT Mobile Application," in Korean Information Processing Society (KIPS), Jeju National University, Jeju Self-Governing Province, Republic of Korea, 2017.

[11] G. Denaro, A. Margara, M. Pezz, and M. Vivanti, "Dynamic data flow testing of object oriented systems," in Proceedings of the 37th International Conference on Software Engineering - Volume 1, Florence, Italy, pp. 947-958, 2015.

[12] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Detecting data-flow errors based on Petri nets with data operations," *IEEE/CAA Journal of Automatica Sinica,* vol. 5, no. 1, pp. 251-260, 2018.

[13] A. Cavarra, "Data flow analysis and testing of abstract state machines." The ICASM, B and Z pp. 85-97, 2008.

[14] P. Jorgensen, *Software testing : a craftsman's approach*, Fourth edition. ed., Boca Raton, Florida: CRC Press, Taylor & Francis Group, 2014.

3D 대화 시나리오 저작도구 개발

고연정 신동춘 지민성 정윤식
주식회사 콜마인
day@thepureun.net, dcshin@thepureun.net, wwwbada910@thepureun.net,
yunsik@thepureun.net

Development of 3D Dialog Scenario Authoring Tool

Yeon-Jeong Ko, Dong-Chun Shin, Min-Seong Jee, Yun-Sik Jeong
Callmine inc.

요 약

본 보고서는 대화 시나리오를 3차원 환경에서 관리할 수 있는 3D 대화 시나리오 저작도구에 관한 것이다. 본 연구에서 대화 시나리오를 구성하는 각 대화문장을 3차원 의미벡터로 표시하였으며, 대화 시나리오 전개를 3차원 상에 가시화하였다. 또한, 대화 시나리오를 입력, 수정, 삭제가 가능하다.

하나의 대화 시나리오는 다수의 문장으로 구성되며, 각 문장은 문장을 구성하는 대표 단어의 의미벡터 합으로 표현된다. 대화 시나리오는 키워드 입력을 통해서 검색할 수 있으며, 검색된 대화 시나리오는 3차원 공간상에 문장을 선으로 연결하여 가시화하였다.

1. 서 론

1.1 연구 배경

최근 인공지능 에이전트, AI 스피커, 각종 스마트 로봇들이 출시되고 있으며, 사람의 말을 이해해서 적절한 응답을 제공하기 위해서 인간의 언어를 기계가 이해하기 쉽게 표현하는 연구가 진행되고 있다.

본 연구는 인간의 말, 즉, 문장을 의미벡터로 표시하고, 대화 전개를 시나리오 기반으로 진행한다는 가정하에 시나리오를 문장 의미벡터의 연속으로 표시해서, 3차원상에 가시화하여, 시나리오가 어떻게 전개되는지 볼 수 있는 도구를 제공하고자 한다.

또한, 시나리오를 구성하는 문장을 삽입하거나 수정, 삭제하는 기능도 제공하여, 효과적으로 대화 시나리오를 관리하도록 하는 것을 목표로 한다.

1.2 관련연구

최근 트위터로부터 일상 대화 시나리오를 수집하는 방법이 연구되었다. [1] 이렇게 수집된 대화 시나리오를 관리하는 도구에 대한 필요성이 대두되고 있다.

2. 본 론

2.1 전체 시스템 구성도

전체 시스템은 아래와 같이 머신러닝서버와 3차원 대화 시나리오 뷰어로 구성된다. 머신러닝서버에서 웹에서 수집한 문장을 대상으로 단어에 대한 3차원 벡터가 모델링되며, 대화 시나리오 문장이 의미벡터로 표시되어 3차원 대화 시나리오 뷰어 상에서 표시된다. 3차원 대화 시나리오 뷰어에서는 시나리오 검색, 시나리오 수정, 시나리오 전개 와 같은 기능을 제공하여, 시나리오를 3차원에서 관리할 수 있도록 지원하게 된다.

대화 시나리오는 다수의 문장으로 구성되는데, 각 문장은 그 문장에서 발생한 단어의미벡터의 합으로 표현하였다. 미리 학습하지 못한 복합명사로 문장이 새롭게 만들어 질 경우, 그 복합명사를 구성하는 단어의 의미벡터합으로 그 복합명사의 의미벡터를 표현한다.

아래 표는 3개의 단어로 구성된 복합명사의 의미벡터를 그 복합명사를 구성하고 있는 각 단어 의미벡터 조합으로 표현한 예이다.

복합명사	x 좌표값	y 좌표값	z 좌표값	단어 분리
인물추고도감	-0.02672	-0.22696	0.762833	도감;인물;추고;
인물풍속도무덤	0.178075	-0.17488	0.65056	무덤;인물;풍속도;
인민민주주의국가	-0.83856	-0.04771	0.406166	국가;민주주의;인민;
인민민주주의제도	-0.68759	-0.21081	0.482514	민주주의;인민;제도;
인민보건체조	-1.10825	-0.34399	-0.42693	보건;인민;체조;
인민유격투쟁	-0.61637	-0.13033	0.127841	유격;인민;투쟁;
인민항쟁운동	-0.5301	-0.14541	-0.10506	운동;인민;항쟁;
인쇄전신부호	0.474588	-0.62625	0.158844	부호;인쇄;전신;
인식논리학	0.692593	-1.03159	0.352226	논리;인식;학;

표 1. 복합명사의 의미벡터

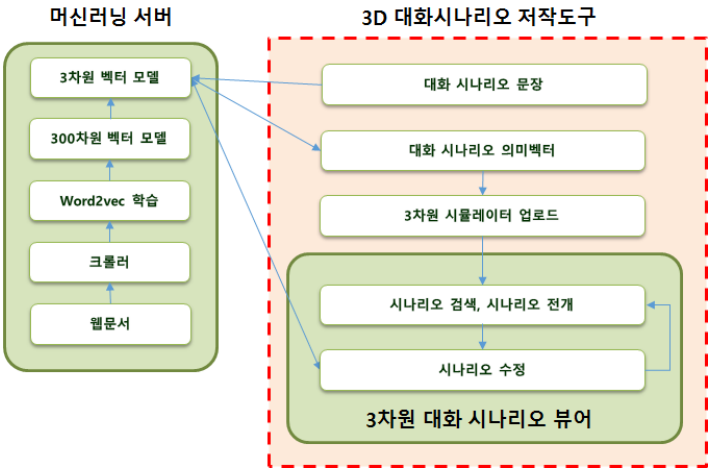


그림 2. 시스템 개요

2.2 차원 압축

기존에 많이 이용되고 있는 Word2Vec모델을 통해 위키피디아에서 수집한 28만개의 단어에 대한 300차원 벡터값을 획득하였다. 또한, 300차원을 3차원으로 표시하기 위해서 PCA(Principal Component Analysis)를 이용하여 차원을 압축하였다.

아래는 R Studio에 300차원 데이터를 입력하여 공분산, 상관행렬을 구해서, 주성분 분석을 통해서 실제 3차원 좌표값을 도출한 화면이다.

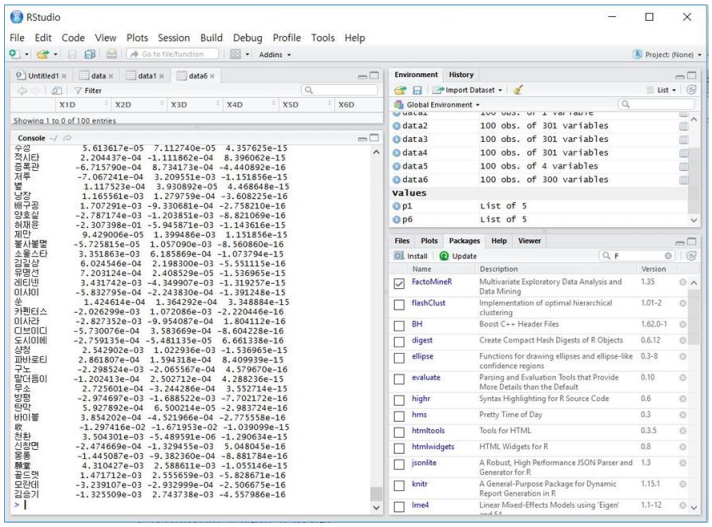


그림 2. R Studio를 이용한 3차원 좌표값 획득

2.3 3차원 대화 시나리오 뷰어

3차원으로 대화 시나리오를 저작, 관리하기 위하여 3차원 시나리오 뷰어를 개발했다. 대화 시나리오를 구성하는 각 문장은 의미벡터의 연결로 표현되며, 각 의미벡터는 3차원 공간에서 고유한 벡터값을 가지고 있다. 이러한 의미벡터가 연결된 하나의 묶음이 하나의 대화 시나리오가 되며, 저작도구를 통해서 생성, 삭제, 편집, 업데이트가 가능하다.

아래 그림은 대화 문장이 공간상에 무수한 점으로 표현된 화면이다. 멀리서 보면 점처럼 보이지만, 실제로는 정육면체 격자모양으로 구성되었으며, 시나리오의 전개 방향이 정육면체의 중앙에서 서로 연결되는 구조로 만들었다.

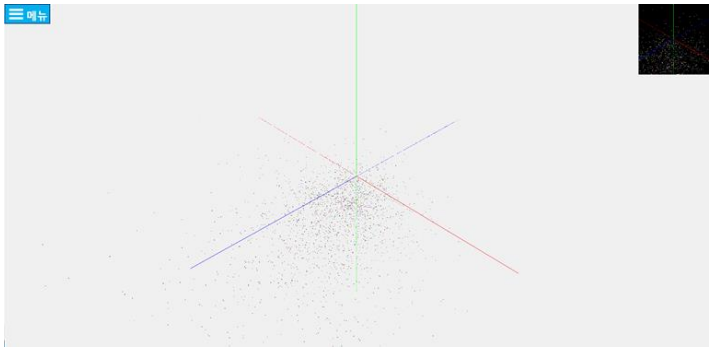


그림 3. 3D 대화 시나리오 뷰어 화면 - 줌아웃

줌인하였을 경우, 점모양이 정육면체로 표시됨을 확인할 수 있다. 우측 상단에는 줌아웃된 3차원 공간을 표시하여, 줌인하더라도, 현재 공간상의 위치를 실시간 확인할 수 있도록 하였다.

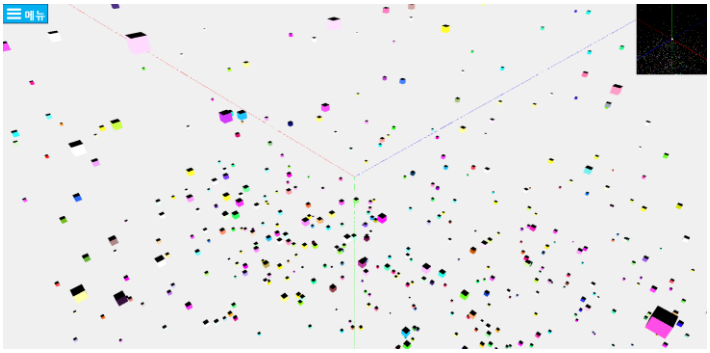


그림 4. 3D 대화 시나리오 뷰어 화면 - 줌인

대화 시나리오를 검색하기 위해서, 검색 키워드를 입력하였을 경우, 유사한 문장 리스트가 좌측에 표시되며, 우측에는 해당 문장이 포함된 전체 시나리오가 표시된다.

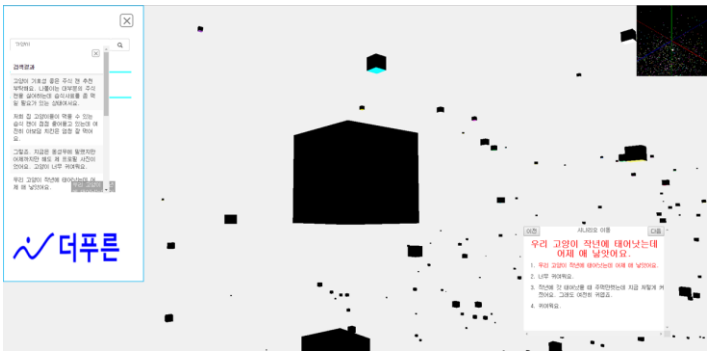


그림 5. 3D 대화 시나리오 뷰어 화면 - 검색 키워드 입력 : 고양이

예를 들자면, “고양이” 키워드로 검색하였을 경우, 아래와 같은 검색결과가 표시된다.

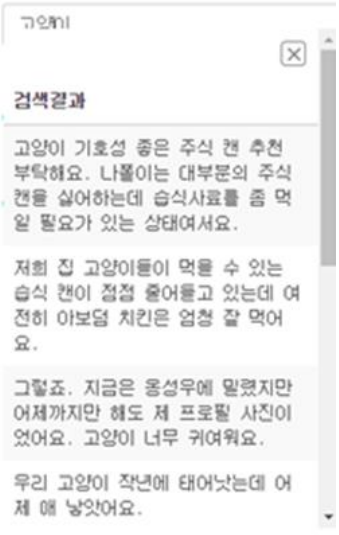


그림 6. 3D 대화 시나리오 뷰어 화면 - 검색된 문장 리

스트

3차원 공간의 시나리오상에서 문장간 이동이 가능하며, 우측 하단 팝업창으로 이동하는 기능을 제공한다.

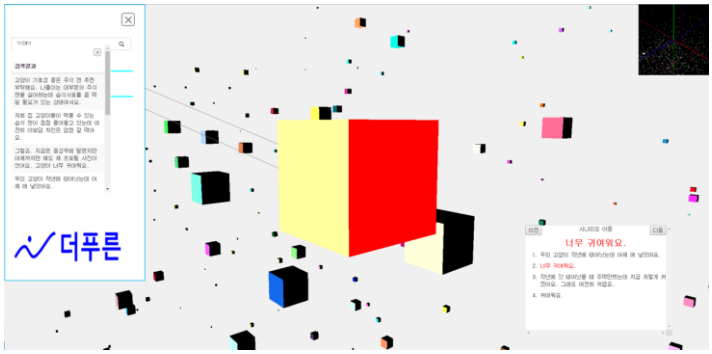


그림 7. 3D 대화 시나리오 뷰어 화면 - 다음 문장 이동

아래 그림은 우측 하단의 팝업창을 확대한 화면이며, “이전”, “다음” 버튼을 통해서 하나의 대화 시나리오 상의 이전 문장과 이후 문장으로 이동이 가능하다.

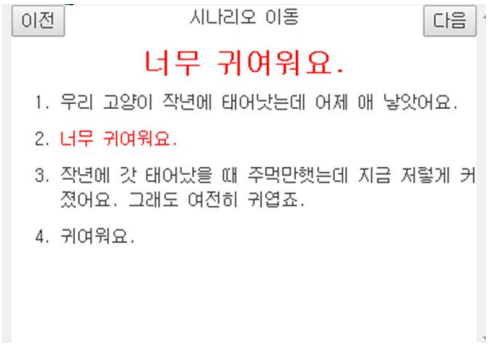


그림 8. 3D 대화 시나리오 뷰어 화면 - 시나리오 이동 화면

대화 시나리오의 각 문장은 선으로 연결되며, 공간상에서 이동상황을 한 눈에 파악할 수 있다.

즉, 1번 문장 -> 2번 문장 -> 3번 문장 -> 4번 문장으로 이동하며, 2번 문장(너무 귀여워요)과 4번 문장(귀여워요)은 의미벡터 간의 거리가 매우 가까우며, 거의 같은 의미를 나타냄을 확인할 수 있다.

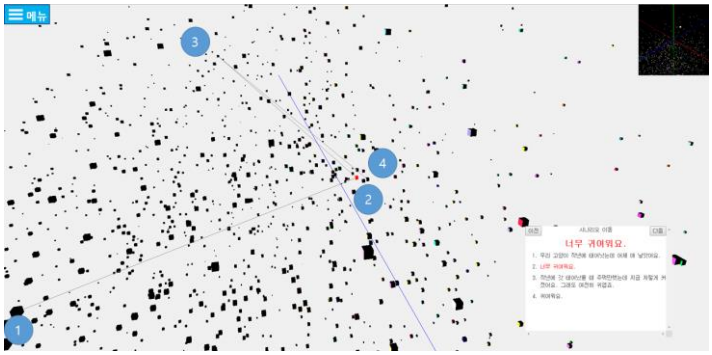


그림9. 3D 대화 시나리오 뷰어 화면 - 시나리오가 공간상에 선으로 표시

2.4 속도 최적화

파일에서 데이터를 읽어서 28만개의 정육면체 큐브를 3차원공간에 렌더링하면 약 41분의 시간이 소요된다. 렌더링 속도를 최적화하기 위해서 3차원 공간을 27등분하였으며, 현재 카메라 위치를 중심으로 해당 영역에 있는 정육면체 큐브만 렌더링하여 렌더링 시간을 17초로 단축하였다. 또한, 모든 큐브에 문장을 표시하면 렌더링 속도가 느려지는 이슈가 발생한다. 이를 해결하기 위해 클릭한 큐브에만 문장이 표시되게 수정하여 속도를 향상시켰다.

3. 결 론

본 연구를 통해서 2차원으로 관리하던 시나리오와 의미벡터를 3차원으로 관리할 수 있게 되었다. 그리고 머신러닝 서버를 이용하여 단어의미벡터 학습이 자동으로 이루어지게 된다. 그 결과 시나리오 관리가 훨씬 편리한 환경을 구축하였다.

4. 사 사

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [2016-0-00563, 자율지능 동반자를 위한 적응형 기계학습기술 연구개발]

5. 참고문헌

[1] 윤재민, 김요한, 이주경, 이중길(2017). “추론기반 능동적 데이터 수집 기술 개발”, 2017년 한국정보과학회 춘계학술대회.

순환 인공 신경망을 이용한 코드 변경 추천

이재준 ○, 강성원, 이선아*
한국과학기술원
{jaejun.lee, sungwon.kang}@kaist.ac.kr
*경상대학교
saleese@gnu.ac.kr

Code Change Recommendation based on Recurrent Neural Networks

Jaejun Lee○, Sungwon Kang, Seonah Lee*
Korea Advanced Institute of Science and Technology
*Gyeongsang National University

요 약

개발자는 소프트웨어 변경 작업을 수행할 때 수정해야 할 필요가 있는 파일을 탐색하는데 많은 시간을 소비한다. 이러한 시간적 비용을 줄이기 위해서 개발자에게 변경해야 할 부분을 추천해주는 변경 추천에 관한 연구들이 수행되었다. 변경 추천을 위해서는 이용한 데이터의 종류에 따라 개정 이력 기반의 접근 방법과 동작 이력 기반의 접근방법이 있다. 본 연구는 동작 이력 기반의 접근방법으로 기존 연구에서 이용한 동작 정보 외에 개발자가 소프트웨어 변경 과정에서 행하는 동작의 순서 정보를 이용하는 순환 인공 신경망 기반의 추천 방법을 제안한다.

1. 서 론

개발자는 소프트웨어 변경 작업을 수행할 때 수정해야 할 필요가 있는 파일을 찾는데 많은 시간을 소비한다[1]. 이런 시간적 비용을 줄이기 위해 개발자에게 수정이 필요한 파일을 추천하는 시스템인 코드 변경 추천 시스템이 고안되었다[2].

변경 추천 시스템은 개정 이력 기반의 변경 추천 시스템과 동작 이력 기반의 변경 추천 시스템으로 나눌 수 있다. 개정 이력 기반의 대표적 연구인 Zimmermann et al.[3]은 버전 관리 시스템에 저장된 개정 이력을 수집하여 변경된 파일 간의 연관 규칙을 마이닝하는 추천 시스템을 제안하였다. 동작 이력 기반의 대표적 연구인 Lee et al.[1]은 Mylyn[4] 데이터에서 수집된 동작 이력을 이용하여 연관 규칙을 마이닝하여 변경 추천에 사용하였다. 변경 추천 시스템은 주로 연관 규칙 마이닝 기법을 사용하며 이 두 가지 방향으로 발전하였지만, 개발자의 동작 순서 정보와 같은 시간적 정보는 활용하지 않았다. 동작 이력 기반의 연구로서 논문 [5]는 개발자의 동작 순서 정보를 활용한 N-그램 모델을 제시하였다. 이 연구는 처음으로 시간적 정보를 활용한 모델이나 연관 규칙 마이닝 기법을 사용했기 때문에 패턴이 일치할 경우에만 추천을 하여 정밀도는 높지만 다른 모델에 비해 추천률이 낮은 약점을 갖고 있다. 연구 [6]은 인공신경망을 도입한 모델을 제시하여 패턴 매칭의 단점을 극복하고 보다 유연한 추천이 발생할 수 있게 하였으나 개발자의 동작 순서 정보인 시간적 정보를 추천에 활용하는 가능성을 탐구하지 않았다.

일반적으로 Mylyn과 같이 개발자의 동작을 추적 및 기록하는 도구에서는 발생하는 여러가지 동작 정보를 시간 순으로 기록한다. N-그램 모델을 제외한 대부분의 변경 추천 기법들은 주로 데이터의 출현 빈도를 추천에 이용하고 있다. 그러나 개발자는 수정해야 할 코드를 결정하기 위해 연관성이 있는 파일을 일반적으로 순차적으로 읽거나 수정하기 때문에, 개발자의 동작 정보로부터 얻을 수 있는 동작 순서 정보와 같은 시간적 정보를 활용한다면 추천 정확도를 높일 가능성이 있다. 그리고 연관 규칙 마이닝 기법 대신 인공 지능에 기반한 학습 모델을 사용한다면 보다 유연한 추천이 발생하여 추천률이 낮아지는 것을 막을 수 있을 가능성이 있다.

본 논문에서는 이러한 기존 연구의 한계점을 극복하기 위해 동작 순서 정보를 이용하여 학습할 수 있는 순환 인공 신경망(Recurrent Neural Network)에 기반을 둔 변경 추천 방법을 제안한다.

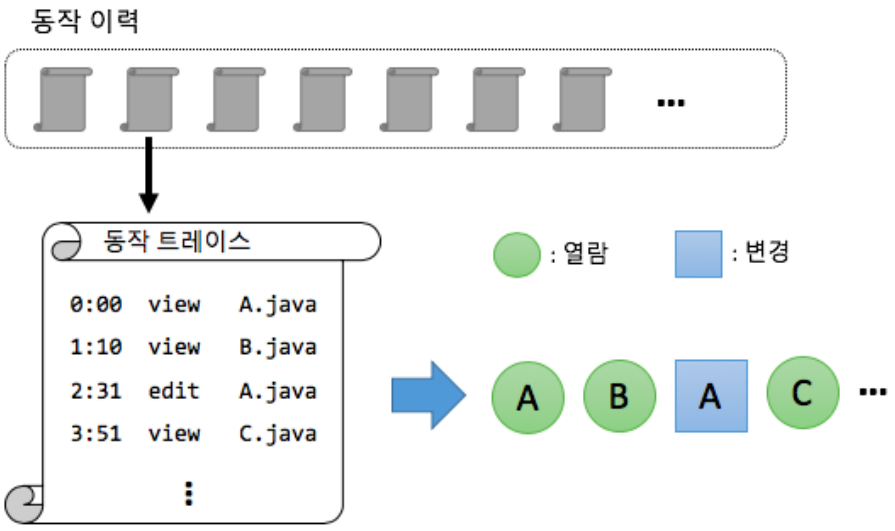


그림 1 동작 이력과 동작 트레이스

2. 순환 인공 신경망 기반의 변경 추천 방법

2.1 동작 이력과 동작 트레이스

순환 인공 신경망 모델을 생성하고 평가하기 위해 동작 이력으로부터 데이터 셋을 구성해야 한다. 동작 이력의 기본 단위를 동작 트레이스라 정의하며, 소프트웨어 진화 작업을 하는 동안 기록된 로그 파일 한 개를 의미한다. 동작 이력은 동작 트레이스의 집합이며, 시간 순으로 정렬되어 있다. 그림 1은 동작 이력과 동작 트레이스의 구성을 보여준다.

2.2 학습 데이터 생성 방법

학습 데이터는 문맥(Context)과 변경 추천 대상으로 구성된다. 문맥은 '추천을 발생시키는 질의문'으로 정의한다[1]. 본 연구에서 제안하는 문맥과 MI에서 제안하는 문맥의 가장 큰 차이점은 개발자의 동작 순서를 유지하는가의 유무이다. 기존 MI의 문맥은 동작들의 집합이기 때문에 순서와 상관없이 문맥을 구성하여 모델을 학습하지만 본 연구에서 제안하는 모델은 슬라이딩 윈도우(Sliding Window)를 적용하여 동작 순서를 유지하며 문맥을 형성할 수 있도록 한다.

그림 2는 임의의 동작 트레이스에서 크기가 4인 슬라이딩 윈도우를 이용하여 시간 순서를 유지한 채, 문맥을 형성하고 해당 문맥에 대한 변경 추천 대상을 쌍으로 묶어 학습 데이터를 생성하는 과정을 보여준다. 변경 추천 대상은 동작 트레이스에서 도출된 문맥 이후에 발생하는 변경 파일로 구성된다.

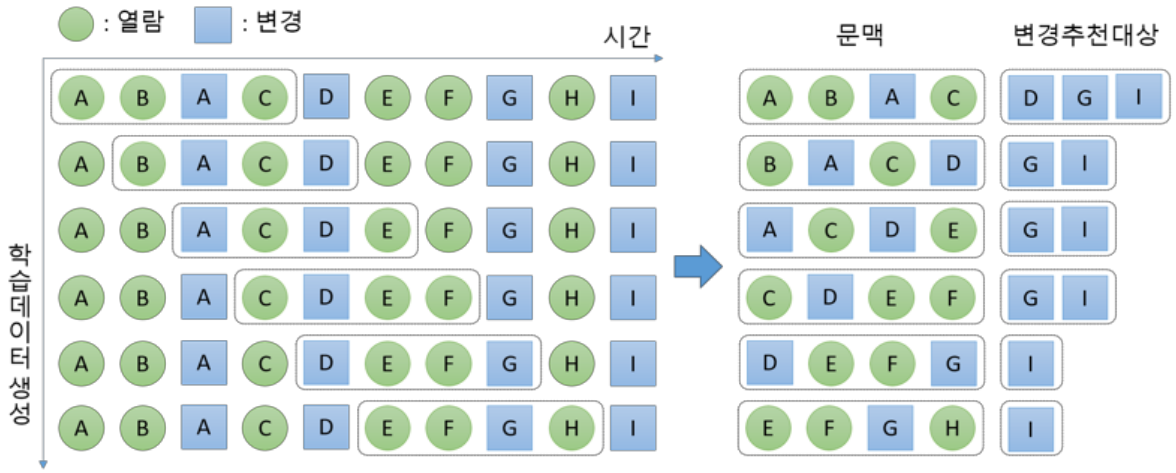


그림 2 학습 데이터 생성 과정

2.3. 순환 인공 신경망 기반의 변경 추천 방법의 상세

모델 생성 과정은 크게 세 부분으로 나눌 수 있다. 첫째, 모델을 생성하기 위한 데이터 입력 과정이다. 전처리 과정을 통해 정제된 학습 데이터를 정수 인코딩(Integer Encoding)을 이용하여 숫자로 변환한 후, 이를 One-Hot 인코딩 기법을 통해 기본적인 인덱싱 작업을 한다. 방대한 데이터의 학습 시간을 줄이기 위해, 인덱싱 된 학습 데이터 셋을 임베딩 레이어(Embedding Layer)를 통해 차원(Dimension)을 축소할 필요가 있다. 예를 들어, 이클립스 플랫폼 프로젝트(Eclipse Platform Project)의 경우, Mylyn에 의해 기록된 열람 및 변경 파일이 6000개 이상 존재하기 때문에 이를 입력 데이터로 바로 사용할 경우, 높은 차원 때문에 학습 시간이 매우 오래 걸리게 된다. 따라서 임베딩을 사용하여 데이터의 특성을 보존하면서 차원을 축소하고, 이를 학습함으로써 결과에 크게 영향을 미치지 않으면서도 학습 시간을 줄일 수 있다.

둘째, 입력 데이터로부터 순환 인공 신경망을 이용하여 출력값을 도출하는 과정이다. 임베딩 레이어를 거친 데이터는 순환 인공 신경망 중 하나인 LSTM(Long Short-Term Memory)[7] 레이어를 통해 모델을 생성하게 된다. 하나의 문맥에 대해서 다수의 변경 추천을 받기 위해, 순환 인공 신경망을 통해 도출한 출력값 중 다수를 선택할 수 있는 멀티 레이블(Multi-Labels) 모델을 구현한다. 이를 위해, LSTM 레이어에서 활성화 함수(Activation Function)로 시그모이드 함수(Sigmoid Function)를 사용한다.

마지막으로 위의 과정을 통해 도출한 출력값과 실제 정답 셋을 비교하여 손실(Loss)을 줄이는 방향으로 학습하며 아담(Adam) 기울기 하강을 사용하여 수행된다.

3. 결론

본 연구는 개발자의 동작 순서 정보를 이용하는 순환 인공 신경망 기반의 변경 추천 방법을 제안하였다. 초기 실험 결과, 개발자가 파일을 수정 시 다음에 변경할 파일을 추천해주는 방식으로 모델을 구성할 때, 이클립스 PDE 프로젝트에서 0.59의 정밀도(Precision)를 보여주었으며, 이는 최신 연구[1]의 0.52에 비해 약 13% 높은 정밀도이다. 이를 통해, 본 연구에서 제안한 시간적 정보를 활용한 모델이 추천 정밀도를 향상시킬 수 있다는 것을 확인하였다. 현재 이클립스 플랫폼 프로젝트, ECF 프로젝트, MDT 프로젝트에 대해 추가 실험을 진행 중이며, 본 연구의 제안 방법과 기존 방법들을 비교 분석할 예정이다.

Acknowledgement

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발 사업의 지원을 받아 수행된 연구임. (NRF-2017M3C4A70 66210)

참고 문헌

[1] Lee, Seonah, Sungwon Kang, Sunghun Kim, and Matt Staats. "The impact of view histories on edit recommendations." IEEE Transactions on Software Engineering 1 (2015): 1-1.

[2] Robillard, Martin, Robert Walker, and Thomas Zimmermann. "Recommendation systems for software engineering." IEEE software 27.4 (2010): 80-86.

[3] Zimmermann, Thomas, Andreas Zeller, Peter Weissgerber, and Stephan Diehl. "Mining version histories to guide software changes." IEEE Transactions on Software Engineering 31, no. 6 (2005): 429-445.

[4] Kersten, Mik, and Gail C. Murphy. "Mylar: a degree-of-interest model for IDEs." Proceedings of the 4th international conference on Aspect-oriented software development. ACM, 2005.

[5] 김태현, 강성원, 이선아, 금창섭, N-그램 모델 기반의 코드 변경 추천 시스템, 제 19회 한국 소프트웨어공학 학술대회, 제19권 제1호 (2017): 108-111.

[6] 이준범, 강성원, 안휘, 정필수, 이선아, "인공신경망을 이용한 코드 변경 추천 시스템," 제 20회 한국 소프트웨어공학 학술대회, 제20권 제1호 (2018): 151-154.

[7] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

시스템 오브 시스템즈 협업 정책 구성을 위한 상향식 기법 제안

B. Zelalem Mihret O, 지은경, 백영민, 배두환
한국과학기술원
{zelalem, ekjee, ymbaek, bae}@se.kaist.ac.kr

A Bottom-up Approach for System-of-Systems Collaboration Policy Formulation

B. Zelalem Mihret O, Eunkyoung Jee, Young-Min Baek, Doo-Hwan Bae
School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

Abstract

Constituent systems (CSs) in System-of-Systems (SoS) need to be coordinated, and somehow guided, in order to achieve a specific goal. Otherwise, interactions in an SoS will be chaotic or useless, even dangerous. Therefore, to coordinate CSs' operations in an SoS, there is a need to put order and governance of CSs' interactions. This can be achieved by implementing a collaboration policy. A collaboration policy is used to influence CSs' behaviors and guide their actions to achieve SoS-level goals. The underlying CSs composing the SoS can be heterogeneous in their semantics. In addition, though CS could have homogenous semantics, they could be of different domains. To deal with such challenges in formulating a collaboration policy, assuming CSs have similar semantic representation using a rule modeling language such as *Semantics of Business Vocabulary and Rules (SBVR)*, we presented a systematic procedure for collaboration policy rule formulation guided by a machine learning method called *Global Vector Model (GloVe)*. *GloVe* is used to extract unique and shared features of CSs composing the SoS under consideration. In turn, these features can be used to influence CSs' behaviors in the collaboration policy formulation.

1. Introduction

Constituent systems (CS) in a System-of-Systems (SoS) need to be coordinated, and somehow guided, in order to achieve a specific goal. Otherwise, the interactions will be chaotic or useless, even dangerous. Therefore, to coordinate systems operation in system of systems, it is required to put order and governance of their interactions. This can be achieved by implementing a collaboration policy. A policy is used to guide CSs' actions towards a desired goal. It provides permission or obligate CS actions to influence their behavior [1].

A policy, in any setting, is used to provide guidance and governance rules to satisfy goals. In most cases of policy definition and modeling, a policy is structurally represented using conditional clauses as condition and respective actions [2]. The idea of a policy in conventional systems, for example Rei, Ponder [3], and Web Service policy [4], considers mainly the purposes of the hosting system (i.e., the system that controls the implementation of the policy). For example, in such cases a policy may command a client to respect and fulfill requirements of the host system before using specific resources. In these three representative cases of a policy, resource ownership and access to resources are the central theme of a policy. However, for SoS-level systems, where CSs are autonomous that they can act and behave unpredictably, a policy is an essential backbone that enable collaboration for common goals. A policy for a SoS is concerned about governing and guiding actions of interacting CSs towards achieving SoS-level common goals. It fulfills this intent by influencing behaviors of interacting CSs.

With this regard, we presented a SoS collaboration policy modeling method (with a hybrid approach), emphasizing the top-down approach, in our earlier research work [5], as shown in Figure 1. The top-down approach focuses on decomposing SoS-level goals into specific policy rules systematically. It extrapolates CSs' behaviors into five constructs including environment, role, constraint, belongingness and dedication that the collaboration policy should consider to influence CSs' behaviors.

In this paper, we will discuss a machine learning approach to extract CSs' behaviors from bottom-up, assuming CSs are represented using a rule modeling language called *SBVR (Semantic of Business Vocabulary and Rules)* [6]. To analyze the SBVR corpus, we propose a similarity analysis procedure guided by *GloVe (Global Vector model)*, which is an unsupervised learning algorithm for obtaining vector representation for words [7].

The remaining part of the paper has the following sections;

*This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government(MSIT) (No. 2015-0-00250, (SW Star Lab) Software R&D for Model-based Analysis and Verification of Higher-order Large Complex System), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1A6A3A11931209).

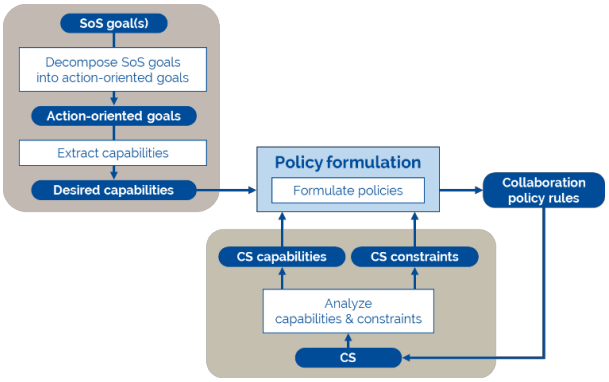


Figure 1. Hybrid approach for SoS collaboration policy formulation

Section 2 presents basic concepts involved in the proposed procedure including SBVR and GloVe, Section 3 presents a procedure for the bottom-up approach, followed by a conclusion.

2. Basic Concepts

In this section, we presented a brief discussion on core concepts and extended techniques for our proposed work. We use MCI response system as illustrative example. MCI response system is one of the representative cases for SoS. CSs that compose MCI response SoS includes Fire and rescue services, Emergency medical services, and Hospitals. We use VeTIS tool to develop the SBVR representation of the constituent systems

2.1. System-of-Systems and Collaboration Policy

2.1.1. System-of-systems (SoS)

Maier [8] describes a SoS based on architecting principles into five characteristics, which include operational independence, managerial independence, evolutionary development, emergent behavior, and geographic distribution. Similarly, Boardman & Sauser[9] describe a SoS focusing on system concepts and compositions. The characteristics include autonomy, connectivity, diversity, emergent and belongingness. Emergent behavior and evolutionary development are considered as the essential and intrinsic characteristics manifested by a SoS [10] [11]. More specifically, modeling and analyzing CSs' dynamic interactions and their unpredictable behaviors that evolve through are unique characteristics defining a SoS.

2.1.2. SoS collaboration policy

Based on our analysis, (1) a collaboration policy primarily puts in place to guide entities' envelop actions instead of internal operations of the entity, (2) a collaboration policy needs to consider capabilities of entities it aims to influence. It focuses on permitted and required behaviors of entities while performing their operations. Envelop actions can be actions that lead to or happen after the entities main actions [5].

2.2. Semantics of Business Vocabulary and Rules (SBVR)

SBVR is a standard established by object management group [6]. The vision of *OMG SBVR* is expressing business knowledge in a controlled natural language reducing ambiguity to be understandable by human and computer systems [12]. Since SBVR has formal logic of representation, business requirements and rules specified using SBVR are machine processable. There are two major advantages of SBVR for collaboration policy making. First, SBVR provides a metamodel for business rules and the vocabulary on which business rules are defined. SBVR specification or the semantic model represents terminological ontology (vocabulary) as a cohesive set of interconnected concepts, not just a list of terms and definition. It can be used to analyze relatedness of CSs to collaborate for a specific SoS-level goal. Second, a semantic model of the SBVR contains behavioral guidance (e.g., policy, rules, etc.) that governs the actions of subject of the terminological ontology. Therefore, the SBVR semantics of the CSs can be source of CS's behaviors, which the collaboration policy aims to influence. The detail description of SBVR constituents is given as follow.

2.2.1. SBVR Business Vocabulary

The main elements of SBVR vocabulary are *concepts* and *fact types*. The vocabulary of SBVR consists of terms, definitions for concepts, and relations between concepts. *Concepts* are of three types, *object types*, *individual concepts* and *verb concepts* [6].

```
firefighter
    Definition: member of fire department who
    perform rescuing task
together
    Definition: in conjunction with; combined;
two-ways-communication
    Definition: operating or permitting operation
     in either of two opposite directions
radiocommunication
    Synonym: two-ways-communication
Firefighter size-ups
Incident-commander conduct situational-size-up
```

2.2.2. SBVR Business Rules

Rules are specified based on the terminology defined in the vocabulary. Their underlying semantic is defined by statements in predicate logic. There are two types of rules.

(1) *Definitional rules* or *structural rules*: the setup of organization is defined by these rules. The structural business rules use two alethic modal operators. They are used in the sense of 'logically necessary' and 'logically possible/impossible'

(2) *Behavioral rules* or *operative rules*: the behavior of an entity is characterized by these rules. It uses two deontic modal of operators.

```
It is obligatory that each Firefighter move together
It is obligatory that Incident-commander inform
Firefighter about environment
```

2.2.3. SBVR Quantifier and Qualifier

A *fact type* is converted into a rule by adding a modal operator, and then *quantifier* and *qualifier*. Additional fact types can also be used as quantifiers and qualifier. To get an idea about quantifiers and qualifiers, we provide the following example.

```
It is obligatory that each Firefighter protect and
rescue savable-live
```

2.3. Global Vector Model (GloVe)

GloVe is an unsupervised learning algorithm for obtaining vector representations for words [7]. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The two main model families (or learning methods) used in earlier studies for learning word vectors are (1) global matrix factorization method, such as latent semantic analysis, and (2) local context window method, such as the skip-gram model and continuous bag of words (CBOW). Both methods have limitation in performance on word analogy, and poor utilization of statistics of a given corpus, respectively. GloVe uses a specific weighted least square model that trains on global word-word co-occurrence counts to make efficient use of statistics.

A SoS can consist of heterogeneous CSs from different domains. Hence, *terms* and *fact types* can be used differently in their SBVR representation. *GloVe* as a method of meaning extraction can be used to identify similarities among SBVR rules. It can also be used to determine synonym of terms in the corpus.

GloVe notation for meaning extraction:

- Let X denote the matrix of word-word co-occurrence count, whose entries X_{ij} tabulate the number of times word j occurs in the context of word i.
- Let $X_i = \sum_k X_{ik}$ the number of times any word appears in the context of word i.
- Let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that word j appear in the context of word i

The general model for word vector learning takes the ratio of co-occurrence probabilities. Noting that the ratio $\frac{P_{ik}}{P_{jk}}$ depending on three words i, j and k.

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{ik}}$$

Where $w \in R^d$ is a word vector and $\widetilde{W} \in R^d$ are separate context word vectors.

3. Bottom-up policy formulation approach

SoS-level goal decomposition into policy rules represents the top-down view of collaboration policy formulation of the hybrid approach, depicted in Figure 1 [5]. The produced rules can be transformed into SBVR form by using manual encoding, or automatically based on the natural language processing technique such as a transformation model proposed in [13].

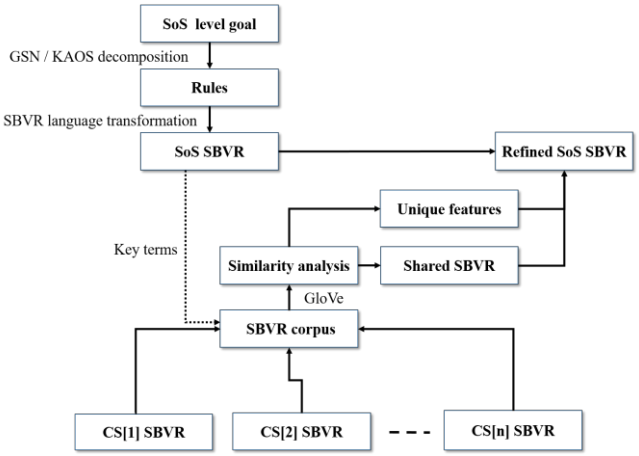


Figure 2. SBVR-based constituent systems collaboration policy formulation

The top-down approach elicits important policy rules in the perspective of SoS-level requirements. However, a collaboration policy should take into account CSs' behaviors it aims to influence [14] [15] . CSs' behaviors can be found embedded in the *concepts* of business terms, *fact types* and *rules*. In our model, we are proposing a systematic procedure of extracting CSs' behaviors encoded in their SBVR semantics. The basic assumption in this approach is that CSs' key terms, internal policy, procedures and requirements are represented using SBVR semantics.

A corpus of SBVR semantics that contain each CS's SBVR and key terms from SoS SBVR (generated by top-down approach) can be established by adding the respective contents. Defining unique and shared feature extraction procedure by extending the GloVe method, SBVR corpus can be analyzed to determine similarities among CSs' SBVRs. The similarity analysis is guided by the key terms from the SoS SBVR as depicted in Figure 2.

The initial SoS SBVR will be refined based on unique feature of CSs and shared SBVR identified from gap and similarity analysis. The refinement mainly aims to incorporate CSs' behaviors in the final SoS collaboration policy.

3.1. Unique Feature Identification

Unique features are those that are extracted from the SoS corpus that fulfils the following conditions,

- The rule containing *necessity claim* key word of SBVR
- The rule containing key terms of SoS SBVR vocabulary
- It appears less than the minimum threshold relative to corpus size, and other rules appearance

The SoS SBVR need to consider such rules to better influence the behavior of the corresponding CSs. This unique feature extraction focuses on identifying specific CSs' behaviors that should be taken into consideration in the collaboration policy formulation.

3.2. Shared SBVR Identification

Shared SBVR contains rules that appear in the majority of CSs' SBVR, and highly related to other rules in each SBVR. If such a rule is not part of the SoS SBVR, then the revised SoS

SBVR need to accommodate them. It provides information about interaction behavior and relatedness of CSs. It provides information about interaction behavior and relatedness of CSs.

$\check{S}::r_s = \{r \mid r \subseteq \bigcup_i^j R_{cs_i}\}$ where $j > \frac{n}{2}$, and

n represents number of constituent system in the SBVR corpus.

To illustrate the use of GloVe guided rule extraction from the SBVR corps; we selected the following three *fact types*, and considered three key terms from MCI goal decomposition. The terms include *communication*, *rescue* and *transport*. Then, we determined μ value that will help to determine relatedness of *fact types* based on the co-occurrence of *fact types* along with the *key terms*.

$$\mu = \frac{P(val|fact_i)}{P(val|fact_j)}$$

where, $P(val|fact_i)$ shows probability of $fact_i$ co-occurance with *key term* val.

$fact_1$	**Incident-commander** *inform* **Firefighter** *about* **environment**
$fact_2$	**Firefighter** *protect* and *rescue* **savable-live**
$fact_3$	**EMT** or **Paramedics** *treat* **Patient** *during* **Transportation**

When the value of μ is closer to integer value 1, it signifies that both *fact types* are equally related to the respective *key term*, otherwise inversely related. This is a possible indicator for shared rules. Inversely related *fact types* indicate that one of them is more related to the *key term*. If μ is greater than 1, the numerator *fact type* is more related to the *key term*. This is a possible indicator for unique rules. To determine the relevance of *fact types*, we need to examine the co-occurrence matrix of *fact types*.

4. Conclusion

The main intention of the collaboration policy is to influence CSs' behaviors so that they can be synergized to fulfill a SoS-level common goal. The intuition behind the bottom-up approach for collaboration policy formulation is to extract CSs' behaviors that the policy aims to influence. In such approach, collaboration policies need to be delivered to the CSs in such a way that CSs can implement them, but CSs can be of heterogeneous domain and use different semantics. Similarly, formulating a collaboration policy from bottom-up does pose challenges of integration and interoperability due to CSs' heterogeneity in domain and semantics. The hybrid approach to formulate SoS collaboration policies is an ongoing research, and it mainly aims to devise a mechanism of policy formulation from both perspectives. With a fundamental assumption that CSs are represented in a common rule model language such as SBVR, we presented a procedure to extract unique and shared features that need to be considered in collaboration policy rules to influence the CSs' behaviors. We proposed a machine learning techniques based on GloVe to extract similar concepts and related rules.

References

[1] M. Hall-May, and T. Kelly, "Defining and Decomposing Safety Poicy for System of Systems," in *SAFECOMP*, Germany, 2005.

[2] G. Chatzikonstantinous, and K. Kontogiannis, "Policy Modeling and Complience Verification in Enterprise Software Systems: A Survey," in *MESOCA'12*, Italy, 2012.

[3] D. Nicodemos, D. Naranker, L. Emil, and S. Morris, "Ponder: A Language for Specifying Security and Management Policies for Distributed Systems," 2000.

[4] V. Kolovski, and B. Parsia, "WS-Policy and Beyond: Application of OWL Defaults toWeb Service Policies," 2014.

[5] Z. Mihret, E. Jee, Y. Baek, and D.-H. Bae, "A collaboration policy model for system of systems," in *SoSE*, Paris, France, 2018.

[6] OMG Object Management Group, "Semantics of Business Vocabulary and Business Rules (SBVR). Version 1.0: Formal Specification.," 2008.

[7] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," 2014.

[8] M. W. Maier, "Architecting Principles for Systems-of-Systems," *INCOSE International Symposium* , 1996.

[9] J. Boardman, and B. Sauser, "System of Systems – the meaning of 'of'," *Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering,* 2006.

[10] D. Fisher, An Emergent Perspective on Interoperation in Systems of Systems, CMU/SEI-2006-TR-003, 2006.

[11] R. Bahsoon, and R. Weinreich, "ECSAW '16: Proccedings of the 10th European Conference on Software Architecture Workshops," in *ACM*, Copenhagen, Denmark, 2016.

[12] N. Lina, S. Tomas, S. Algirdas, S. Edvinas, and A. Linas, "VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models," in *ICIST'10*, 2010.

[13] S. Ramzan, I. S. Bajwa, I. U. Haq, and M. A. Naeem, "A model transformation from NL to SBVR," in *ICDIM*, 2014.

[14] W. Yun, D. Shin, and D.-H. Bae, "Mutation Analysis for System of Systems Policy Testing," in *SESoS'17*, e Buenos Aires, Argentina, 2017.

[15] B. Agusdinata, Exploratory modeling and analysis: A promising method to deal with, TU Delft University of Technology,, 2008.

BIMM에 기반한 소프트웨어 조직의 비즈니스 혁신성 가이드를 위한 적용사례

박영식*, 김영철
홍익대학교 소프트웨어공학연구실
{park12160422*, bob}@selab.hongik.ac.kr

Applied practices for guiding business innovation of Software Company based on Business Innovation Maturity Model (BIMM)

Young-Sik Park*, R. Young Chul Kim
SE Lab, Dept. of Software & Communications engineering , Hongik University

요 약

앞으로의 소프트웨어 비즈니스는 더 자주, 더 훨씬 요구충족적인 신상품이나 서비스를 개발 해야 할 것이다. 또한 다가온 4차산업혁명 시대에 있어 소프트웨어 조직의 적응력 향상을 위해 혁신 도입이 필수적이다. 이는 미래의 SI업체 흥망을 가를 수 있는 요소가 될 것으로 본다. 문제는 기업의 비즈니스 대한 혁신성을 알 수 없다는 점이다. 이를 해결하기 위해, 일리노이공대(IIT)의 Dr. Carlson의 BIMM 모델을 적용하여 비즈니스 혁신성 제시하고자 한다. 평가 및 개선의 가이드을 위해, 1) BIMM의 단계별 활동과 2) 증상 분석 및 도출 하고, BIMM의 평가기준표을 통해 국내 블록체인 관련 기업에 대한 비즈니스 혁신성의 평가 사례를 보인다.

1. 서 론

소프트웨어 기업의 비즈니스 혁신에 대한 평가 및 가이드가 필요하다. 새로운 방법을 찾고, 좀 더 풍부하고 미래를 예측할 수 있는 혁신전문가로 유도한다. 그래서 모든 조직의 브레인들은 고객에 대한 서비스 정책 파트에 배치된다. 혁신에 대한 도전은 존재하기 마련이다. 혁신로드맵은 기관이 이익을 내고, 플러스 성장의 유지를 위하여 깔끔하고 탄탄한 진행에 대한 가이드와 평가를 필요로 한다[1]. 그 가이드와 평가를 위하여 BIMM의 단계별 활동과 증상을 분석하여 카테고리 별 추출에 이은 단계별 변화를 살펴볼 필요가 있다. 아울러 국내 블록체인기업인 "㈜****시스템"사에 BIMM의 평가기법을 적용하여 비즈니스혁신성 평가를 실시한 사례를 기술한다.

2. BIMM의 모델

먼저 비즈니스 혁신성 평가에 앞서 BIMM의 각 단계의 활동과 증상 분석을 통하여 카테고리 별로 추출한 단계별 변모를 표1과 표2에서 언급하고 있다.

2.1 BIMM의 단계별 활동 분석

SUPPORT/ORGNIZATION/EMPLOYEE 카타고리에서 보면 SPORADIC 단계에서 NURTURED 단계[1]까지는 꾸준하게 발전하는 변모가 보인다. 그러나 SUSTAIN단계에서는 그 카테고리가 존재하지 않음을 알 수 있다. 즉, 지원/조직/종업원에 관련되는 사항의 활동은 NURTURED 단계에서 종결된다. Goals 카테고리에서는 SPORADIC단계의 원가적인 측면에서 시작하여 자산유지, 포트폴리오 형성, 이익과 성장, 마지막 SUSTAINED 단계의 기대 매출에 대한 예산과 목표로 변모한다

2.2 BIMM의 단계별 증상 분석

EMPLOYEE/LEADER/ORGANIZATION카타고리는 SPORADIC 단계에서 시작하여 MANAGED 단계에서 종결된다. 이는 앞에서 기술된 활동 카테고리와 매칭되는 부분이 있음을 볼 수 있다. 반면에 IMPROVED카타고리는 1,2단계를 완성한 다음 단계인 MANAGED 단계부터 나타나서 SUSTAINED 단계에서 최종 목표인 투자자의 최대 투자자본수익률 (ROI:Return On Investment)를 달성하게 된다.

표1. BIMM 단계의 활동 분석 및 도출(카타고리별)

구 분	I,Sporadic	II,Idea	III,Managed	IV,Nurtured	V,Sustained
Support	혁신을지원하는 비공식리더쉽	혁신을 지원하는 공식적 리더쉽			
Organization		아이디어관리를 위한 비공식적 조직	CEO에게 직보고하는 혁신 부서장이 있는 공식적 조직	이익 또는 성장 향한 혁신을 목표로 하는 모든 부서의 설립	
Employee	고용자의 간헐적 참여	일부 고용자에 대한 훈련	적용를 위하여 지능적이고 혁신방법속에서의 제품 개발자 훈련		
Culture	혁신의 필요성 인식	소수의 간부프로젝트	고용자에 대한 리스크 시도와 경험을 허용	적용을 위해 모든 직원에게 우수성과 혁신에 대한 교육을 제공.	수익성장을 유지할 계획과 리뷰
		아이디어에 대한 인센티브	아이디어 관리 프로세스가 정의되고 문서화	모든 부서에서의 혁신 경쟁 보장	기업 혁신지수 수립
		포상과 인정 시스템의 적용	MBA를 구체화하는 혁신포트폴리오 생성, 열린 혁신과 조직의 혁신, 시장과 혁신드라이브 기술	혁신 인덱스지수 구축	혁신 포트폴리오의 매년 정기적 업데이트
			포트폴리오에 대한 자원 공급 완수	혁신의 탁월성 포상	새로운 성장 기회 및 새로운 일자리 파악
			열린 그리고 조직적혁신에 대한 혁신 관리계획 수립		매3-5년의 신플랫폼 소개
Goals	원가에 초점	지능적 자산 유지 훈련	혁신포트폴리오 형성	이익과 성장	기대하는 수익성장을 위한 예산과 목표

표2. BIMM 단계의 증상 분석 및 도출(카타고리별)

구 분	I,Sporadic	II,Idea	III,Managed	IV, Nurtured	V,Sustained
Employee/ Leader/ Organization	계획되지 않은 직원 혁신	아이디어 관리 리더	조직도, 직무 기술 및 고안된 가시적 혁신공간위에서의 혁신 기능을 담당하는 최고 혁신 책임자		
Recognition	지역화된 인식			새로운 혁신에 대한 업계의 인식	미디어에서의 기업 인지도
				직원당 아이디어 수 증가 그리고 더 많은 CEO 추천	
Culture	혁신 리더쉽의 강 열림	아이디어 관리 분야에서 교육받은 리더십	새로운 파생 상품 또는 변형 혁신의 상용 출시	대부분의 부서의 새로운 솔루션 활동, 프로세스, 제품 또는 비즈니스 모델 수준 혁신	전사적인 혁신의 내부 및 외부적 영향.
	한계 재무성과	단기 및 장기 혁신 전략	아이디어 관리 소프트웨어 구현		비즈니스 목표 달성 직원/고객 만족
		모든 수준의 참여에 대한 보상	아이디어 관리의 효과 측정		
		전반적으로 배포된 아이디어 관리	시장에서 수익성 있는 혁신		
		아이디어에서 진화하는 혁신 프로젝트			
		실험 실패에 대한 허용 오차			
Improved			설계 및 개발 프로세스의 향상된 성능	혁신 목표 달성을 위한 성과는 검토되고, 보고되며, 분기별로 소통	투자자를 위한 ROI 향상
					예측 가능성을 갖는 혁신 프로세스 및 그 성과 향상

3. BIMM의 평가 방법론

조직의 적합성을 평가하는 2가지 항목은 BIMM의 규격과 단계의 수준이다. 평가의 주 목표는 혁신 가속을 위한 적합성의 수준을 이해하고, 평가하고 그에 대한 기회도를 평가하는 것이다. 수치적 평가[1-6]는 올바른 수행이 된 후에 업데이트될 수 있는 요구사양의 구현레벨을 평가는 것이다. 행위평가와 점수할당에는 고려하는 3가지 요소가 있다. 각 기술은 3가지 항목에 있는 "%"로 매겨진다. 접근, 적용, 결과로 언급하며, 1) 접근은 방법론의 성숙도와 문서화, 2) 적용은 조직간의 구현 (범위)수준을 말하고, 3) 결과는 전략과 방법 또는 접근이 잘 행하여지는 지를 말한다. 표3에서는 평가기준표를 보여주고 있다.

4. 평가 사례 및 결과

표3의 평가기준표에 근거한 평가를 국내 블록체인업체인 ㈜****시스템에 적용한 사례를 소개한다. ㈜****시스템은 "일하기 좋은 SW 전문 기업"중 전자신문사장상 사내문화 및 소통부문최우수상 수상과 정보통신산업진흥원(NIPA)으로부터 소프트웨어 프로세스(SP) 품질 인증 2등급을 획득한 기업으로 본 평가의 대상으로 적합하다고 선택하였다. 즉 평가는 평가기준표의 평가항목에 대한 현장방문과 설문 (5회)을 통하여 실시하였다. 평가의 결과는 일반적 업무부문에서는 각 부문별 우수한 평가가 있었다. 혁신부문에서 1) 비공식적 혁신 리더쉽이 있으며, CEO의 혁신필요성 인지단계, 2) 혁신조직 (비집중)은 없으며 계획적이지 않은 혁신 시도, 3) 혁신을 위한 제반 사항을 공식화하려는 직전 단계로 볼 수 있다. 평가 결과는 혁신성 평가의 기준으로는 초기단계인 레벨1 결과를 얻었다. 이를 통해 혁신성의 지속적 향상을 위한 정확한 혁신성 도입과 요소 사항등을 가이드 할 수 있었다. 그 평가에 대한 결과를 표4에서 살펴볼 수 있다.

표3. 평가기준표[1]

Rating	Approach(A)	Deployment(D)	Results(R)
(0-20)	비공식 접근	혁신관리프로세스를 위한 준수의 비인지	즉흥적 혁신의 결과
(21-40)	접근의 반응	혁신관리프로세스에 대한 약간의 깨우침과 간헐적 준수	혁신관리프로세스의 효과가 규칙적으로 평가되지 않는다
(41-60)	접근공식시스템의 안정	혁신관리프로세스의 주적응 단계	혁신관리프로세스 효과가 존재하고 그 증빙이 나타난다
(61-80)	지속적 개선이 강조됨	시스템에 대한 완전한 준수와 지속적 업데이트	전체혁신관리시스템의 효과가 증명되고 혁신의 목표가 달성된다
(81-100)	최상의 달성도	완전 준수, 지속 업데이트, 높은 효율의 프로세스	고객의 높은 만족도, 지속적 혁신의 개량이 증명되고, 혁신의 목표가 초과한다.

표4. BIMM level determination(****시스템)

Level	Range of Score
0	1 - 10
1	11 - 30
2	31 - 50
3	51 - 70
4	71 - 90
5	91 - 100

단계	Max	Min. to Qualify
Sporadic	15	12
Idea	15	12
Managed	20	16
Nurtured	25	20
Sustained	25	20

Actual
9
10
11
11
12

Step 1. Calculate total score for the assessment = 53.
Step 2. Determine initial qualified BIMM level (QBL) = 3.
Step 3. Identify the lowest qualified stage without jump (LQS) = None.
Step 4. Calculate the average of remaining stages above LQS (ARS) = 10.6.
Step 5. Estimate final BIMM stage score: LQS Upper Limit + ARS = 10+ 10.6 = 20.6
Step 6. Determine the final qualified BIMM Level (QBL) = 1.

5. 결론

BIMM의 단계를 카테고리 별로 분석하고, 평가기준표에 근거한 평가를 국내 기업에 적용하여 보았다. 이는 소프트웨어 조직의 발전과 미래의불확실성 에 대비할 수 있는 지름길로 가이드 할 것이다. 또한 현재까지 실시되고 있는 평가방식은 CEO 또는 CFO의 면담과 설문, 내부적 보고서 및 서류를 기준으로 실시되는 방식이다. 향후에는 평가방식과 평가기준표에 있어 우리나라 소프트웨어 조직의 실정에 더 적합한 모델을 연구하여 비즈니스 혁신성을 지속적으로 향상시킬 수 있기를 기대한다.

참 고 문 헌

[1] Praveen Gupta and Brett E.Trusko, “Global Innovation Science Handdbook”, Mc Graw HILL Education, 2014.

[2] 김기두, 류동국, 김영철 “TMM향상을 위한 테스트 프로세스 성숙도 체크리스트 연구”, 한국정보과학회, 추계학술발표대회 논문집, Vol.31, No.2, pp.487-489, 2004.

[3] 김기두, 박병호, 김영철 “테스트프로세스 향상을 통한 테스트 성숙도 모델(TMM)에 관한 연구”, KCSE, 한국소프트웨어공학 학술대회, Vol. 1, no.1, pp.155-161, 2005

[4] 김기두, 신석규, 김영철, “적용 사례를 통한 소프트웨어 테스트 평가모델 분석”, KCSE, Vol. 8, no. 1, pp.365-369.

[5] 김기두, 김영철, “테스트 프로세스 개선 모델 TPI next)을 통한 테스트 성숙도 모델 확장에 관한 연구”, 정보처리학회, Vol. 19, No. 1, pp.1243-1246, 2012.

[6] 박영식, 김영철 “4차산업시대에 대비한 “비즈니스혁신성” 측정 모델 소개”, ICT플랫폼학회, 추계학술대회논문집, Vol.6-2, No.9, pp29-32,2018.

위치 분석 기반 통합 클라이언트-서버 프로그램 컴파일 방법 구현

김가영 [1], 최광훈 [1], 창병모 [2]
전남대학교 전자컴퓨터공학과 [1] 숙명여자대학교 컴퓨터과학과 [2]
kirayu15@gmail.com, kwanghoon.choi@jnu.ac.kr, chang@sookmyung.ac.kr

An Implementation of Compilation Based On Location Analysis for Unified Client-Server Programs

Gayoung Kim[1], Kwanghoon Choi[1] , Byeong-Mo Chang[2]
Dept. Electronics and Computer Engineering, Channam National University[1]
Dept. of Computer Science, Sookmyung Women's University[2]

요 약

본 논문은 원격 함수 호출 위치를 분석하여 통합 RPC 프로그램을 클라이언트-서버 프로그램으로 분리 컴파일하는 방법을 구현한 내용을 소개한다. 웹 서비스는 가장 널리 사용되고 있는 플랫폼이지만, 이 서비스를 제공하기 위해 다양한 언어를 사용해야 한다는 단점이 있다. 이 문제점을 해결하기 위해 통합 프로그래밍 언어들이 제시되었다. 그 중 RPC 계산법은 위치를 지정하는 확장된 람다 계산법을 도입하였지만, 위치에 대한 유추 과정이 없어 컴파일이 매우 복잡하다. 본 논문은 이를 단순화시킨 이론을 직접 구현함으로써 이론을 뒷받침하며, RPC 프로그램을 클라이언트-서버 프로그램으로 자동 분리 컴파일 하는 과정을 설명한다. 또한 Http 통신을 이용한 런타임 시스템으로 동작을 확인한다.

1. 서 론

WWW(World Wide Web)는 인터넷에 연결된 환경이라면 어느 누구나 접근할 수 있는 공간으로 가장 널리 사용되고 있는 플랫폼이다. 이 플랫폼에 파일 전송, 원격 접속, 와이즈(WAIS) 등의 다양한 서비스를 제공하는 것을 웹 서비스라 말한다. 이 서비스를 구축하기 위해서, 개발자는 HTML로 웹 페이지의 구조를 작성하고, CSS로 HTML 페이지를 형식화 해주며, JavaScript로 클라이언트와의 상호작용을 위한 동적 기능을 추가한다. 또한 다양한 데이터를 서버에 저장하기 위해 SQL과 같은 언어를 사용해야 하며, 데이터를 구조화시켜 전달하기 위해 JSON을 이용한다. 이처럼 다양한 프로그래밍 언어를 사용한다는 것은 웹 프로그래밍을 어렵게 하며, 테스팅을 복잡하게 만든다.

이러한 문제점을 개선하기 위해 Cooper와 Wadler는 RPC 계산법(calculus) [1]을 제안하였다. 각 함수에 실행할 위치를 지정하도록 람다 계산법(lambda calculus)을 확장한 것이다. 프로그래머는 RPC 계산법으로 통합된 단일 프로그램으로 서버와 클라이언트 측의 코드를 작성할 수 있다. 이 단일 프로그램은 RPC 계산법 컴파일러를 통해 자동으로 클라이언트 코드와 서버 코드로 분리된다.

[그림 1]은 RPC 계산법을 사용한 예제 프로그램이다. 이 프로그램은 클라이언트에서 실행할 함수 getCreds와 서버에서 실행할 함수 auth를 포함한다. getCreds 함수 정의(lam)에 ^c를 붙이고, auth 함수 정의에 ^s를 붙여 각 함수에 실행할 위치를 지정한다. 이 예제 프로그램을 클라이언트에서 실행하면 먼저 12번째 줄의 서버 함수 auth를 호출한다. 서버에서 6번째 줄의 클라이언트 함수 getCreds를 호출하여 사용자 아이디와 패스워드를 입력 받는다. getCreds 함수의 형식 인자 prompt는 실인자 "Enter name, pwd: " 문자열을 전달받아 print 함수를 통해 사용자에게 이 문자열을 보여주고, read 함수로 사용자로부터 입력을 받아 서버측 지역변수 creds로 반환한다. 7번째 줄 if 문에서 creds가 문자열 "ezra:opensesame"와 비교하여 두 문자열이 같다면 "the secret document"를, 같지 않다면 "Access denied"를 클라이언트에 반환한다.

```
01: getCreds = lam^c prompt.
02:   let y = print prompt in
03:     read
04:
05: auth = lam^s x.
06: let creds =
      getCreds "Enter name, pwd: " in
07:   if creds == "ezra:opensesame" then
08:       "the secret document"
09:   else
10:       "Access denied"
```

```
11:
12: main = auth ()
```

그림 1 RPC 프로그램 예제

Choi와 Chang [2]은 함수 호출에서 위치 정보를 타입으로 파악하여 기존 RPC 계산법의 복잡한 구현 방법을 단순화 시킨 컴파일 방법을 제안하였다. 이 타입 기반 컴파일 방법이 정확함을 증명하였지만 본 논문에서 실제로 구현해보임으로써 그 증명을 뒷받침하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 통합 클라이언트-서버 프로그래밍 방법과 관련된 연구를 소개한다. 3장에서는 타입 기반 위치 정보를 활용한 RPC 프로그램의 컴파일 방법에 대한 상세 내용을 설명하고, 4장에서 결론을 맺는다.

2. 관련연구

통합 프로그래밍 언어에 대한 기존 연구는 다음과 같다. RPC 계산법[1]은 람다 계산법에 위치 정보를 추가함으로써 통합된 단일 프로그램으로 서버와 클라이언트 코드를 작성할 수 있다. 이를 기반으로 만들어진 언어가 Links[3]로, 데이터베이스 쿼리 최적화, 연속성, 메시지 전달 및 동시성, XML 프로그래밍을 통합한 함수형 통합 웹 프로그래밍 언어이다.

Ur/Web[4], Hop[5]은 통합 웹 프로그래밍 언어로, 클라이언트에서 서버 함수를 호출할 수 있지만, 서버에서 클라이언트 함수를 호출할 수 없는 비 대칭형 통신만 제공한다. 서버에서 클라이언트의 함수를 호출하기 위해서는 외부 라이브러리를 이용해야 하기 때문에, 해당 언어에서 이와 관련된 구문(Syntax)을 제공하지 않는다.

Cooper와 Wadler의 연구[1,3]에서는 서버 상태를 유지하지 않기(Stateless) 때문에, 반복적으로 클라이언트와 서버 간의 호출이 있을 경우 통신상의 오버헤드가 발생할 수 있다. 데이터베이스 쿼리와 같이 Stateless 프로그램에서 인코딩하기 어려운 프로그램도 존재하기에 Choi와 Chang[2]은 이를 해결할 방법으로 Stateful 람다식과 위치 정보를 파악하여 RPC 계산법의 구현을 단순화시킬 컴파일 방법을 제안하였다. 또한, 다른 언어에서는 클라이언트에서 서버의 함수를 호출하는 비 대칭형 통신을 지원하지만, Choi와 Chang의 연구에서는 클라이언트에서 서버를 호출할 수 있고 서버에서 클라이언트 함수를 호출할 수 있는 대칭형 통신을 지원한다.

본 논문에서는 Choi와 Chang의 연구에서 제안한 이론을 뒷받침하기 위해 Parser, 위치 유추 알고리즘, 컴파일러를 구현했으며, HTTP 기반의 런타임 시스템을 구현하여 프로그램이 실제로 동작함을 확인한다.

3. 타입 기반 위치 정보를 활용한 RPC 프로그램 컴파일 방법

본 논문에서 제안하는 타입 기반 RPC 프로그램 컴파일 방법을 설명한다. [그림 2]는 RPC 프로그램이 클라이언트-서버 프로그램으로 분리되는 과정을 도식화한 것이다. 첫 번째 단계에서는 프로그래머가 작성한 RPC 프로그램의 타입 유추(type inference)를 통해 함수를 실행해야하는 위치를 결정한다. 두 번째 단계에서는 타입 정보를 이용하여 RPC 프로그램에 나타난 각 함수 호출이 지역인지 원격인지를 판단하여 컴파일 변환한다. 세 번째 단계에서는 클라이언트 프로그램과 서버 프로그램으로 분리한다.

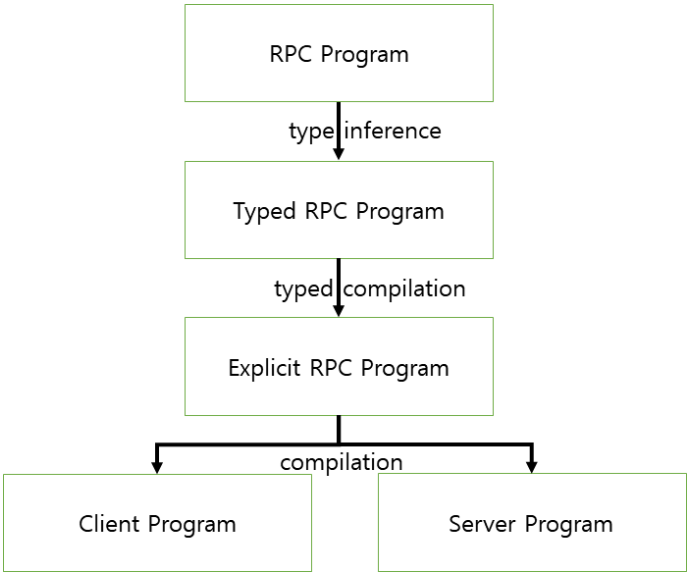


그림 2 클라이언트-서버 프로그램으로 분리하는 과정

앞으로 1장에서 제시한 [그림 1]의 RPC 프로그램 예제를 이용해 설명한다.

3.1 타입 유추(Type Inference) 과정

첫 번째 단계에서는 프로그래머가 작성한 RPC 프로그램의 타입 유추(type inference)를 통해 함수를 실행해야하는 위치를 결정한다. 원격 함수 호출과 지역 함수 호출을 구별하기 위해 이 과정이 필요하다.

[2] 논문에서 위치 정보를 갖는 함수 타입을 다음과 같이 제안하였다.

t1-loc->t2

t1은 함수의 인자 타입이고, t2는 함수의 결과 타입이다. 그리고 loc는 c 또는 s이고, c는 클라이언트, s는 서버를 가리킨다. 이 타입을 갖는 함수는 해당하는 위치(loc)에서 실행됨을 의미한다.

예를 들어, [그림 1]의 1번째 줄의 getCreds는 문자열을 인자로 받고 문자열을 반환하는 함수이고, 'lam^c'로 작성하여 클라이언트에서 실행하도록

정의되었기 때문에 String-c->String의 함수 타입을 갖는다. 5번째 줄의 서버 함수 auth는 void 인자를 받아 문자열을 반환하기 때문에 Void-s->String의 함수 타입을 갖는다.

이와 같은 방식으로 타입 유추 과정을 거치면 [그림 3]와 같은 결과를 얻을 수 있다. 이 과정에서 유추한 타입을 볼드체(Bold)로 구분하여 표시하였다. 각 변수의 타입은 변수 선언에서 ':' 다음에 온다. 예를 들어, 'getCreds: String-c->String = …'는 getCreds 함수의 타입이다. '^c^' 또는 '^s^'는 호출할 함수가 실행될 위치가 클라이언트 또는 서버임을 나타낸다. 예를 들어, 서버에서 클라이언트 함수를 호출하는 코드 'getCreds ^c^ "Enter name, pwd: " '의 ^c^는 getCreds 함수가 실행되어야 할 위치가 클라이언트임을 나타낸다. 클라이언트에서 서버 함수를 호출하는 코드 'auth ^s^ ()'의 ^s^는 auth 함수가 서버에서 실행되어야 함을 나타낸다.

프로그래머가 작성한 RPC 프로그램에서는 함수 정의에만 위치를 지정할 뿐, 함수 호출 코드에 위치 정보를 지정하지는 않는다. 타입 유추 과정에서 이 정보를 자동으로 찾아낸다.

```
getCreds: String-c->String =
  lam^c prompt: String.
    let y: Unit = print ^c^ prompt in
      read
auth: Void-s->String =
  lam^s x: Unit.
    let creds: String =
      getCreds ^c^ "Enter name, pwd: " in
      if creds == "ezra:opensesame" then
        "the secret document"
      else
        "Access denied"

main: String = auth ^s^ ()
```

그림 3 Typed RPC Program

3.2 타입 기반 위치 정보를 활용한 컴파일 과정

두 번째 단계에서는 타입 정보를 이용하여 RPC 프로그램에 나타난 각 함수 호출 'f arg'가 지역인지 원격인지를 판단하여 컴파일 변환한다. 구체적으로, 클라이언트에서 클라이언트 함수를 호출하면 지역 함수 호출로 판단하고, 서버 함수를 호출하면 원격 함수 호출로 판단한다. 서버에서 클라이언트 함수를 호출하면 원격 함수 호출로 판단하고, 서버 함수를 호출하면 지역 함수 호출로 판단한다. 이 내용을 다음의 [표 1]로 요약할 수 있다.

표 1 위치에 따른 함수 호출 구분

	클라이언트 함수	서버 함수
클라이언트에서 호출	(1) 지역 함수 호출	(2) 원격 함수 호출
서버에서 호출	(3) 원격 함수 호출	(4) 지역 함수 호출

[표 1]로 요약된 함수 호출 종류로 판단 내린 다음, 컴파일 변환을 다음과 같이 한다. (2)번의 경우, 'f arg' 형태의 함수 호출을 'Req f arg' 형태의 원격 함수 호출로 변환한다. (3)번의 경우, 'f arg' 형태의 함수 호출을 'Call f arg' 형태의 원격 함수 호출로 변환한다. 나머지 (1)번과 (4)의 경우 'f arg' 함수 호출의 형태를 유지한다. 위에서 설명한 컴파일 변환을 [표 2]와 같이 요약할 수 있다.

표 2 함수 호출 방법에 따른 컴파일 변환 방법

컴파일 변환	컴파일 전	컴파일 후
(2)	`f arg`	`Req f arg`
(3)	`f arg`	`Call f arg`
(1), (4)	`f arg`	`f arg`

예를 들어, [그림 3]의 'auth ^s^ ()'가 (2)에 해당하는 컴파일 변환을 적용한다. 적용한 결과는 아래와 같다.

컴파일 전	컴파일 후
`auth ^s^ ()`	`Req auth ()`

'getCreds ^c^ "Enter name, pwd: "'는 (3)에 해당하는 컴파일 변환을 적용할 수 있다. 적용한 결과는 아래와 같다.

컴파일 전	컴파일 후
`getCreds ^c^ "Enter name, pwd: "`	`Call getCreds "Enter name, pwd: "`

'print ^c^ prompt'는 (1)에 해당하는 지역함수 호출이다. 따라서 컴파일 변환에서 그대로 둔다.

컴파일 전	컴파일 후
print ^c^ prompt	print prompt

[그림 3]에서는 (4)에 해당하는 경우는 없지만, 위의 (1)번의 경우와 같이 그대로 두는 컴파일 변환을 적용한다.

[그림 3]의 예제를 입력 받아 두 번째 단계 컴파일 변환 과정을 거치면 [그림 4]와 같은 결과를 얻을 수 있다. 컴파일 변환 과정 전후의 가장 큰 차이점은 원격 함수 호출을 Req와 Call로 명시적으로 표시한 것이다. Req로 표시된 원격 함수 호출은 클라이언트에서 서버 함수를 호출하고, Call로 표시된 원격 함수 호출은 서버에서 클라이언트 함수를 호출하는 것을 구분한 것이다.

```
getCreds = lam^c prompt.
```

```
  let y = print prompt in
    read

auth = lam^s x.
  let creds =
    (Call getCreds "Enter name, pwd: ") in
    if creds == "ezra:opensesame" then
      "the secret document"
    else
      "Access denied"

main = (Req auth ())
```

그림 4 RPC Program with Req and Call

3.3 클라이언트-서버 프로그램으로 분리

세 번째 단계에서는 클라이언트 프로그램과 서버 프로그램으로 분리한다. 컴파일 단계를 거친 프로그램은 여전히 하나의 프로그램이기 때문에, 클라이언트-서버 프로그램으로 분리해야 한다.

프로그램 내에 존재하는 함수가 정의된 위치에 따라 분리된다. 즉, 클라이언트 위치로 지정된 함수는 클라이언트 측의 함수 리스트에, 서버 위치로 지정된 함수는 서버 측의 함수 리스트로 분리된다.

[그림 4]의 예제 프로그램을 입력 받아 앞에서 설명한 분리 컴파일 과정이 끝나면 [그림 5]와 같은 클라이언트 함수 리스트와 서버 함수 리스트를 출력한다. 클라이언트 함수 리스트에서 메인 프로그램은 main이다. [그림 5]의 예제에서 함수를 표현하는 클로저를 사용한다. 클로저(Closure), 'Clo f {fv1, fv2, …, fvn}'는 함수 코드 f와 환경(Environment)이다. 환경은 함수 코드에서 참조하는 자유 변수(free variable), fv1, fv2, …, fvn의 값을 저장한 자료구조이다.

```
Client function list
main = (Req auth ())
getCreds = (Clo _gf1 {read, print})
_gf1 = {read, print} lam^c prompt.
  let y = print prompt in
    read
_gf2 = {getCreds} lam^c z1.
  let y1 = getCreds z1 in Ret y1

Server function list
auth = (Clo _gf3 {getCreds})
_gf3 = {getCreds} lam^s x.
  let creds = (Call (Clo _gf2 {getCreds})
                    "Enter name, pwd: ")
  in
    if creds == "ezra:opensesame" then
      "the secret document"
    else
      "Access denied"
```

그림 5 Stateful Client-Server Program

3.4 클라이언트-서버 통신 구현 방법

Http 프로토콜을 기반으로 클라이언트 프로그램과 서버 프로그램 간의 원격 함수 호출을 지원하는 런타임 시스템을 구현하였다. Req 원격 함수 호출은 Http 프로토콜의 요청(Request)으로 구현하고, Call 원격 함수 호출은 Http 프로토콜의 응답(Response)로 구현하였다. 지역 함수 호출은 통신 방법과 무관하게 일반 함수 호출로 구현하였다.

4. 결 론

이 논문에서 원격 함수 호출 위치를 분석하여 통합 RPC 프로그램을 클라이언트-서버 프로그램으로 자동 분리 컴파일하는 방법을 구현하였다. 분리된 클라이언트 프로그램과 서버 프로그램 간의 통신을 지원하기 위한 Http 프로토콜 기반의 런타임 시스템을 구현하였으며, 분리 컴파일 된 프로그램이 잘 동작함을 확인했다.

5. 참고문헌

[1] E. Cooper, P. Wadler, "The RPC Calculus," Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, September 2009.

[2] K. Choi, B.M. Chang, "A Theory of RPC Calculi for Client-Server Model," To Appear in Journal of Functional Programming, Cambridge University Press, 2019.

[3] E. Cooper, S. Lindley, P. Wadler, J. Yallop, "Links: Web Programming Without Tiers," International Symposium on Formal Methods for Components and Objects(FMCO '06), November 2006.

[4] A. Chlipala, "Ur/Web: A Simple Model for Programming the Web," Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Language, January 2015.

[5] M. Serrano, G. Berry, "Multitier Programming in Hop," Communications of the ACM, Vol. 55, No. 8, pp. 53-59, 2012.

선형 SVM과 Random Forest를 이용한 DNA 메틸화 영역 예측

이충근O, 한경숙
인하대학교 컴퓨터공학과
lchmo4@inha.edu, khan@inha.ac.kr

Validation using Linear SVM and Random Forest for prediction of DNA methylation regions

Chungkeun LeeO, Kyungsook Han
Dept. of Computer Engineering, Inha University

요 약

DNA 메틸화은 DNA의 사이토신에 메틸기가 결합된 현상을 말한다. DNA 메틸화은 생명현상을 이해하는데 중요한 기작으로 작동한다. DNA 메틸화는 노화, 배아의 발달, 암 등에 상관관계가 있다. 기존의 서열에서 RNA또는 DNA조각들에서 제한된 범위에서 결합부위를 예측하는 한계점이 있다. 본 논문에서는 염색체 단위에서 메틸화가 된 DNA 서열을 예측방법의 검증을 논의한다. 메틸화된 특성을 이용하여 예측모델을 개발하였고, DNA서열의 특성 그리고 상대방의 단백질 특성 등의 특징으로 표현하였다. 모델화된 데이터는 약 1:2의 양과 음의 데이터 비율을 가진다. Random Forest 기법과 선형 SVM을 활용하여 교차검증을 하였다. 모든 검증은 5, 7, 10 교차검증을 실행하였다. 이 예측모델의 가장 높은 성능은 Random Forest와 선형 SVM에서 79.63%, 78.78% 정확도를 보였다.

1. 서 론

ChIP-seq와 같은 첨단 실험 기법을 이용하여 메틸화된 DNA 서열 데이터가 많이 알려지기 시작하였고, 단백질과 결합하는 DNA 또는 RNA 서열을 예측하는 모델링 기법도 개발되었다. 예를 들어 TFImpute [1]와 DeepBind [2]은 서열들은 비결합 서열을 shuffle method방식을 통하여 서열들을 추출하였고 이와 관련하여 딥러닝 기법을 이용한다. DeepBind는 각각의 모델을 분할하여 DNA 및 RNA 서열들을 예측한다.

메틸레이션은 발현과정을 억제하는 인자로 알려져 있다. 메틸레이션 과정에서 영향을 미치는 인자는 전사과정을 일으키는 요소 중 하나인, 프로모터에 있다. 이 프로모터에 메틸화 과정이 일어나 발현을 억제하는 것이다. 기존의 작은 조각의 DNA서열을 통하여 예측을 하는 것은 염색체의 다른 서열을 참고 못하고 잘못된 예측을 하는 경우가 존재하였다. DNA는 염색체에 존재하는 작은 조각들로 나머지 염색체 위치에 있는 서열들도 참고하는 것이 정확하게 메틸레이션된 서열들을 예측하는데 도움이 된다. 근래에 들어서 랜덤한 서열이나 shuffle method방식을 통한 서열기반의 예측기법들이 많이 나왔다. 이러한 이유는, 기존의 실험기법들이 DNA의 길이가 짧은 서열들을 추출하는데 초점이 맞춰져 있었고, 이 서열들을 분석하는 툴들도 이와 유사한 서열들을 통계학적인 방법을 통하여 서열들을 추출하였다. 최근까지도, DNA 서열들을 가공할 때, 실험기법으로 나온 서열을 분석하여 이와 관련한 비결합서열들을 생성한 후, 예측모델을 만들었다. 이것은 염색체의 다른 서열들을 참고하여 결합부위를 성공적으로 예측하지 못하는 데이터 사용의 한계점을 드러낸다. 서열기반의 예측은 조각된 서열들을 주로 사용하고 분석하여 연구들이 주로 진행되었다.

iDNA-Methy [3] 기법은 메틸레이션된 서열을 분석하여 이와 관련하여 포지티브 서열과 네거티브 서열에 관하여, 분할한 후, 메틸레이션된 서열을 분석하였다. iDNA-Methy기법은 기존의 제공되어진 서열을 통해서 모델을 만든다. 그리고 DNA 서열을 통해 예측할 수 있다. DeepBind 기법은 ChiP-seq의 데이터를 활용하여 단일 모델을 만들어 예측하는 기법이다. 이 기법은 심층 신경망을 이용하여 모델을 만든다. 실험서열에 대해서 비교적 정확한 예측을 한다. DeepBind는 각각의 실험서열에 대하여 개별적인 모델을 만들고, 길이가 긴 서열 (100bp)에 대한 예측성능을 제공하지 않는다. 또한, DeepBind기법은 서열에 대한 특징을 알기 어려워 데이터에 의존하는 단점이 있다. 이 연구는 메틸화된 서열들의 특징을 발굴하여, 길이가 긴 서열에 대한 단일 모델을 만들고 기계학습방법을 적용하였다.

DNA 메틸화를 서열단위로 예측하는 문제와 관련해서, iDNA-Methy 기법은 기존의 메틸화 서열은 분석한 후, 훈련과정을 거친다. 이때 기존의 서열의 결합 및

비결합서열에 대해 조정작업을 거친다. 이것은 제한된 범위의 서열을 통해서 학습하기 때문에 메틸화 된 서열이외의 다른 유전체 위치의 서열에 대해 포함을 못 시키는 단점이 있다. 추가적으로, 메틸화 되는 서열에 대해서 작은 양의 데이터를 활용한 문제가 있다.

이에 따라, 본 논문에서는 실제 생화학적 실험을 통해 메틸화된 서열을 염색체 단위로 추출하여 메틸화된 서열과 메틸화 되지 않은 서열로 나눠 서열단위로 모델을 만들어 메틸화를 예측할 수 있는 모델을 만들고 이를 검증하였다.

2. 데이터 및 특징

2. 1 DNA 메틸화 데이터 와 단백질 데이터

DNA 메틸화 데이터는 CHIP-Cor, CHIP-Peak, FindM을 통하여 나온 메틸화된 서열들을 추출하였다 [4]. 그리하여 포지티브 데이터로서 메틸화된 DNA 결합부위 데이터로 인간 2번 염색체에 대해서 DNA 서열 18,724개를 추출하였다. 이에 따라, 본 논문에서는 실제 생화학적 실험을 통해 메틸화된 서열을 염색체 단위로 추출하였다. 메틸화된 서열들 중에서 메틸화된 서열과 결합하는 단백질의 수는 총 8개이다. 메틸화된 DNA와 결합하는 단백질 서열은 NCBI (https://www.ncbi.nlm.nih.gov/protein/)의 사이트에서 얻었다. 네거티브 데이터로서 참조 인간 유전체 GRCh37/hg19에서 2번 염색체에 대하여 40,230개 (포지티브 데이터의 2.148배)의 DNA와 비-메틸화하는 영역을 선택하였다. 참조로 사용된 인간 유전체는 메틸화 DNA 영역보다 많은 비-메틸화 영역을 포함하고 있다. 다음 <표 1>에서는 서열중복제거 후의 데이터의 개수가 확인 가능하다.

<표 1> 중복 제거 후 남은 학습 데이터의 서열 개수.

	메틸화된 염기가 포함된 서열	메틸화된 염기가 포함되지 않은 서열
H2AZ	1497	5037
H3K4me2	1409	5576
H3K4me3	1713	0
H3k9me3	0	7886
H3k27me3	618	5072
H3K36me3	9838	5059
H3k79me3	2984	5213
PolII_RPB4	665	6387

2. 2 서열의 선택 및 가공

위의 서열들은 길이 100으로 선택된 서열이다. 위 서열들을 1차적으로 CD-HIT-EST [5]을 이용하여 서열 유사도가 80%이상인 데이터는 전부 삭제하였고 이 데이터를 윈도우 슬라이딩 기법을 활용하여 서열의 길이를 11로 분할하여 재 추출했다. 이렇게 함으로써, 전체 포지티브 서열의 개수는 1,795,485이다. 네거티브 서열의 개수는 4,038,702이다.

2. 3 특징의 선택 및 표현

본 연구에서 DNA 메틸화 서열을 예측하기 위하여 사용한 특징들은 다음과 같다. 유전체 DNA의 GC 정보 비율로서, 각각의 각 염기의 값을 계산하였다. A는 0.3, C는 0.2, G는 0.2, T는 0.3의 값이다. Partner unique feature은 아미노산(단백질) 염기의 위치에 따른 normalization 값의 합이다. DNA mass값은 DNA 염기 하나의 molecular mass값으로 A는 331.2, C는 307.2, G는 347.2, T는 322.20이다. DNA pka는 DNA 염기 하나의 molecular pka값으로 A는 3.5, C는 4.2, G는 10.8, T는 9.90이다. Triplet composition값으로 3쌍의 염기(AAA…TTT) 각각의 수의 합을 계산한 값입니다.
본 연구에서는 서열 길이 11이며, DNA, 단백질 서열을 기준으로 특징을 표현하였다. 첫번째 특성의 개수는 11개이며, 두번째 특성의 개수는 21개이며, 세번째 특성의 개수는 11개이며, 4번째 특성의 개수는 11개이며, 5번째 특성의 개수는 64개이다. 총 특성의 개수는 118개의 특징을 사용하였다 [6, 7].

3. 기계학습의 성능 평가

3. 1 SVM과 Random Forest

SVM은 LIBLINEAR [8]을 활용하여 모델을 만들고, 그 후 교차검증을 실행한다. 선형 SVM에서 아래의 2가지 식을 활용하여 검증하였다.
L2-regularized L1-loss SVC solves (L2L1):

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \left(max\left(0, 1 - y_i w^T x_i\right)\right), \quad (1)$$

L2-regularized L2-loss SVC solves (L2L2):

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \left(max\left(0, 1 - y_i w^T x_i\right)\right)^2. \quad (2)$$

Random Forest는 아래의 알고리즘을 활용하여 모델을 만들고, 그 후 교차검증을 실행하였다. Random Forest는 weka [9]을 활용하였다.

Algorithm 1: Random Forest

Input: dataset Tree = (a, b), number of trees nt, number of random feature f
Output: RF, a set of grown n trees
Initialize RF;
For I = 1 to nt **do**
  Special treatment of the first element of line I;
  Tree' <- boostrap(Tree);
  Tree' <- trainDT(Tree', f);
  addTreetoRF;

end

3. 2 성능 지표

본 연구에서는 정확한 성능을 측정하기 위하여 총 6가지의 성능 지표를 이용하였다. 그 성능은 accuracy (ACC), sensitivity (SN), specificity (SP), positive predictive value (PPV), negative predictive value (NPV), matthews correlation coefficient (MCC)로 식 (3)~(8)로 정의된다.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (4)$$

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (5)$$

$$\text{PPV} = \frac{TP}{TP+FP} \quad (6)$$

$$\text{NPV} = \frac{TN}{TN+FN} \quad (7)$$

$$\text{MCC} = \frac{(TP\times TN)-(FP\times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (8)$$

위의 수식에서 true positive (TP)는 결합한다고 예측된 서열이 실제로 결합하는 경우의 수, true negative (TN)는 결합하지 않는다고 예측된 서열이 실제로도 결합하지 않는 경우의 수, false positive (FP)는 결합한다고 예측된 서열이 실제로는 결합하지 않는 경우의 수, false negative (FN)는 결합하지 않는다고 예측된 서열이 실제로는 결합하는 경우의 수를 의미한다.

3. 3 교차검증

5, 7, 10 교차검증을 통한 학습 모델의 성능을 확인하기 위하여 학습 데이터의 특징들을 인코딩 하여 학습 모델을 만들었으며, 학습 모델을 이용하여 5, 7, 10 교차검증을 수행하였다. <표 2, 3, 4>는 weka의 Random Forest기법을 이용하여 5, 7, 10 교차검증을 나타낸 결과이며, 포지티브와 네거티브의 비은 1대 2.249365649이다. Random Forest의 파라미터 bagSizePercent와 batchsize는 7로 설정하고 검증을 실행하였다. random forest의 나머지 파라미터는 기본적인 값들을 사용하였다. 선형 SVM의 파라미터는 기본적인 값들을 사용하였으며, L2-regularized L2-loss support vector classification (dual)과 L2-regularized L1-loss support vector classification (dual)을 사용하였다. 성능지표 값들을 활용하여 아래에 표에 제시한다. 기계학습 방법 및 검증하는 방법 <표 2, 3, 4>에 따라서 성능의 차가 발생할 수 있다는 것을 나타낸다.

<표 2> 5 교차검증

	SN	SP	ACC	PPV	NPV	MCC
RF	0.60	0.88	0.79	0.69	0.83	0.50
SVM (L2L2)	0.15	0.94	0.70	0.56	0.71	0.17
SVM (L2L1)	0.61	0.86	0.78	0.66	0.83	0.49

<표 3> 7 교차검증

	SN	SP	ACC	PPV	NPV	MCC
RF	0.60	0.88	0.79	0.69	0.83	0.50
SVM (L2L2)	0.34	0.97	0.77	0.83	0.76	0.43
SVM (L2L1)	0.61	0.84	0.77	0.64	0.83	0.47

<표 4> 10 교차검증

	SN	SP	ACC	PPV	NPV	MCC
RF	0.60	0.88	0.79	0.69	0.83	0.50
SVM (L2L2)	0.24	0.95	0.74	0.73	0.74	0.31
SVM (L2L1)	0.60	0.85	0.77	0.64	0.82	0.46

3. 4 기존 연구와의 예측성능 비교

본 연구에서 개발한 기계학습 모델은 기존의 메틸화 서열의 예측 방법인 iDNA-Methy와 비교하기 위한 실험을 하였다. 공평한 성능 비교를 위하여 기존에 논문에서 제시된 값과 비교하였다. 우리의 논문은 서열단위로 구분이 안되고 염기단위로 예측성능을 검증한다. 아래의 <표 5>는 논문에서 제시한 성능과 우리가 제안한 성능의 지수를 비교하였다.

우리가 사용한 기계학습 방법 (RF)이 iDNA-Methy와 비교하여 종합적인 성능수치인 accuracy에 대해서 약간 높지만, TP 와 TN에 초점을 둔 성능에서는 iDNA-Methy에 비해 부족한 성능을 보인다.

<표 5> 다른 모델과의 비교

	Our model		
Accuracy	Sensitivity	Specificity	MCC
79.63	60.33	88.21	50.62
	iDNA-Methy		
Accuracy	Sensitivity	Specificity	MCC
77.49	61.25	90.33	54.71

4. 결론

본 연구에서 DNA 메틸화 서열의 정보와 단백질 정보를 모두 고려하여 메틸화가 가능한 영역을 찾는 모델에 대해 2가지의 기계학습 기법으로 검증하였다. 예측 모델을 위하여 기계학습 모델을 구현하였으며, 사용한 특징으로서 유전체 DNA의 GC정보, Partner

unique feature, DNA mass, DNA pka, Triplet composition을 사용하였다. 데이터로는 포지티브 및 네거티브의 비율이 약 1:2정도로 데이터셋을 만들어서, 성능평가로서 5, 7, 10 교차검증 및 다른 예측성능과도 비교를 진행하였다.

본 연구 결과는 최근 쏟아져 나오는 Chip-seq의 대량의 데이터를 사용하여 메틸화 서열을 예측모델을 검증하였고, 직접적으로 도움이 되는 정보를 제공한다. 향후 관련 분야의 연구에 상당한 도움을 줄 수 있을 것으로 보인다.

5. 감사의 글

본 연구는 미래창조과학부의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2017R1E1A1A03069921).

6. 참고 문헌

[1] Qin, Qian, and Jianxing Feng. "Imputation for transcription factor binding predictions based on deep learning." *PLoS computational biology* 13.2 (2017).

[2] Alipanahi, Babak, et al. "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning." Nature biotechnology 33.8 831 (2015).

[3] Liu, Zi, et al. "iDNA-Methyl: Identifying DNA methylation sites via pseudo trinucleotide composition." *Analytical biochemistry* 474 69-77 (2015).

[4] Ambrosini, Giovanna, et al. "The ChIP-Seq tools and web server: a resource for analyzing ChIP-seq and other types of genomic data." BMC genomics 17.1 938 (2016).

[5] Li, Weizhong, and Adam Godzik. "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences." Bioinformatics 22.13 1658-1659 (2006).

[6] 이현진, and 한경숙. "상호작용 상대방을 고려한 단백질-결합 RNA 염기 예측 기법." 정보과학회논문지: 컴퓨팅의 실제 및 레터 19.5 229-234 (2013).

[7] 최혁진, 최성욱, and 한경숙. "Random forest 를 이용한 단백질에서의 DNA 결합 부위 예측." 정보과학회논문지: 소프트웨어 및 응용 39.7 515-522 (2012).

[8] Fan, Rong-En, et al. "LIBLINEAR: A library for large linear classification." Journal of machine learning research 9.Aug 1871-1874 (2008).

[9] Venkatesan, N., and G. Priya. "Study and Implementation of Random Forest Algorithm with WEKA Tool." *International Journal of Emerging Research in Management and Technology* 365-370 (2015).

순환신경망을 활용한 맥락 데이터 기반 사용자 인증 기법

남상진, 김순태, 신정훈
전북대학교 소프트웨어공학과
sangjin930@gmail.com, stkim@jbnu.ac.kr, shinjh@jbnu.ac.kr

Context Aware Authentication Using Recurrent Neural Networks

Sangjin Nam, Suntae Kim, Jung-Hoon Shin
Dept. of Sotfware Engineering, Chonbuk National University

요 약

본 연구에서는 모바일 기기에서 수집 가능한 맥락 데이터를 기반으로 한 순환신경망을 활용하여 사용자 인증을 수행하고자 한다. 사용한 맥락 데이터는 위치, 통화기록, 애플리케이션 사용기록이 있으며 이 데이터들을 시간에 따른 발생 빈도를 집계하여 순환신경망 모델의 입력으로 한다. 생성된 순환신경망은 사용자나 외부 애플리케이션의 인증 요청이 발생하면 요청 시점으로부터 최근 일부의 데이터를 사용하여 인증 점수를 산출한다. 이러한 인증 방식은 사용자의 제어 없이 수행 가능하기 때문에 사용자가 모바일 기기의 애플리케이션을 사용할 때 사용성이 증가하며, 확장성 또한 높기 때문에 다른 사용자 인증 기법들과 함께 사용 가능하다는 이점이 있다. 우리는 실험을 통하여 해당 기법이 본래의 사용자와 타인을 구분하는데 66%의 정확도를 제공한다는 점을 확인하였다.

1. 서 론

모바일 기기의 사용이 증가와 스마트폰의 대중화로 인하여 애플리케이션 시장의 규모가 커지고 다양한 애플리케이션이 등장하고 있다. 또한 많은 애플리케이션이 서비스를 제공하기 위해서 사용자 인증 단계를 통해 사용자를 구분하고 있다. 인증에 사용되는 기존 기법은 패스워드, OTP(One Time Password), 생체 인증 기법과 공인인증서를 통한 인증 등이 있다.[1,2,3] 대표적으로 기존의 금융 서비스에서는 공인인증서를 통한 사용자 인증이 필수적으로 진행되었다. 하지만 공인인증서의 의무사용이 폐지되면서 다른 사용자 인증 기법이 필요하게 되었으며[4], 앞서 언급한 기존의 인증 기법, 패스워드와 OTP는 인증을 위해 사용자가 계속에서 기억해야 하거나, 메시지 송수신의 시간과 비용이 발생한다는 점에서 한계가 있다. 또한 별도의 사용자 제어가 필요하여 사용에 있어서 불편함이 있다. 이런 문제를 해결하기 위해서 최근에서는 생체 인증 기법이 넓게 쓰이고 있으며, 대표적으로 지문 인식과 얼굴 인식이 쓰이고 있다. 하지만 지문 인식은 습도나 상처 등의 요인으로 인해 인식이 되지 않거나 인식률이 좋지 않은 문제가 있으며, 얼굴 인식의 경우에는 안경의 유무나 주변이 어두워 인식이 되지 않는 등, 경우에 따라서 인식이 불가능한 일이 발생한다는 문제가 있다.

이러한 문제들을 해결하고자 사용자의 맥락 데이터를 활용한 암시적 인증 기법이 연구되고 있다.[5,6,7,8] 이 연구들은 사용자의 행동 패턴을 분석하여 기존의 사용자 패턴과 인증 요청시의 패턴을 비교 분석하여 인증 유무를 판별하는 기법에 대한 것으로, 사용자의 시간별 위치나 기존에 저장된 행동에 대한 발생과 저장되지 않은 행동에 대한 발생 빈도를 활용해 조건부확률이나 가설검정 등을 활용하여 인증 유무나 인증 점수를 산출하는 방식이다.

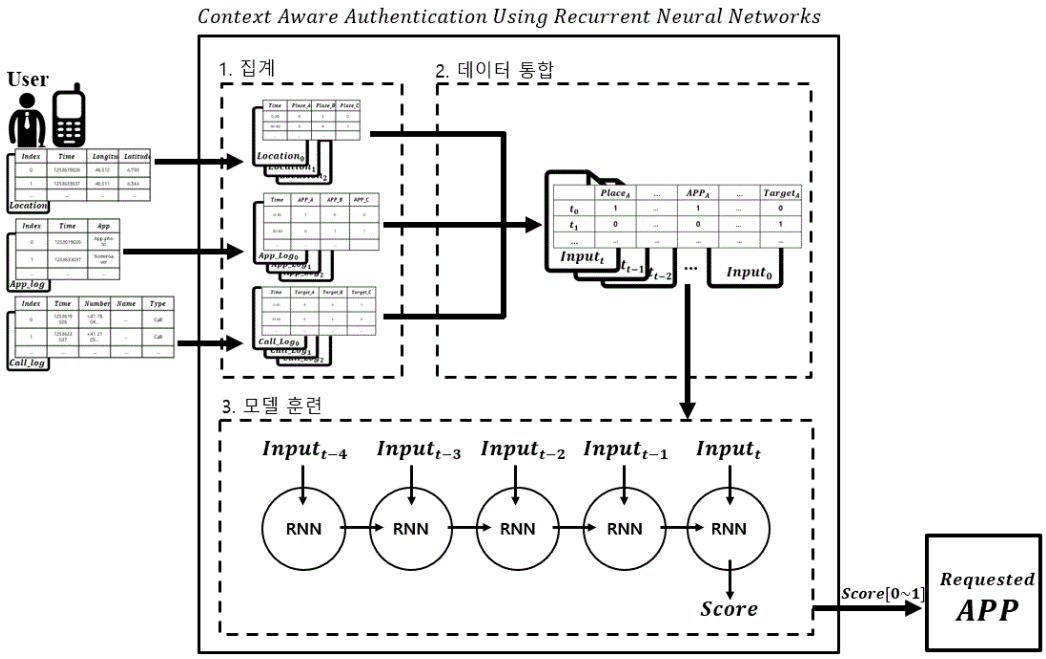


그림 1. Context Aware Authentication Using Recurrent Neural Networks

그러나 기존의 암시적 인증 기법들은 기존 데이터 전체가 아닌 인증 요청 시점의 일부 데이터를 비교하는 것으로 진행되거나, 평균만을 비교하는 등, 맥락 데이터 전체의 특징을 포함하지 않는 모델이나 공식을 사용한다. 따라서 본 연구에서는 데이터의 전체적인 특징을 인증 기법에 반영하고자 한다. 제안하는 기법은 세 단계를 통해 인증 점수를 산출한다. 첫 번째 단계에서 사용자의 종류별 데이터를 일정 기간동안 발생한 행동의 빈도로 집계하고, 두 번째 단계에서 각 데이터를 같은 기간끼리 이어 붙여 입력으로 사용한다. 세 번째 단계에서는 이렇게 만들어진 데이터를 사용하여 인공신경망의 한 종류인 순환신경망(Recurrent Neural Networks)[10]을 학습시켜 과거 모든 데이터에 대한 모델을 생성한다. 생성된 모델은 향후 인증 요청이 발생하면 요청 시점을 기준으로 일정 기간의 데이터를 첫 단계, 둘째 단계와 같은 집계 방법을 통해 입력 데이터로 형태를 바꾼 후에 인증 점수를 평가한다. 이 평가 모델은 데이터 마다 가중치를 수정하기 때문에 모델 하나가 과거의 모든 정보를 표현할 수 있으며, 학습 이후에 기존의 데이터를 계속해서 저장하고 있을 필요가 없기 때문에 효율적이다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 기존 연구와 그 문제점에 대해서 요약하고, 3장에서는 제안한 순환신경망을 사용한 사용자 인증 기법에 대해서 설명한다. 4장에서 제안한 기법을 실험을 통해 검증하며 5장에서 마무리한다.

2. 관련 연구

Elaine Shi의 암시적 인증 기법 연구에서는 맥락 데이터로 문자기록, 위치기록, 브라우저기록 등을 사용하여 각 맥락 데이터가 기존에 저장되어 있는 긍정적인 경우와, 처음 저장되는 부정적인 경우로 분류하였다. 또한 분류된 데이터들의 발생 시간 차이를 계산하여 시간에 따른 인증 확률로 산출하였다. 이러한 방법은 사용자의 과거 맥락 데이터의 일부와 최근 데이터를 비교하여 계산하기 때문에 일부 데이터에 의존하며, 과거 모든 데이터를 반영하지 못한다는 문제가 있다.[6,7]

Eiji. H의 연구는 장소에 따른 인증 기법의 적절성에 대하여 설명하고 있다. 이 연구는 사용자의 기기 제어를 통한 데이터(Active data)와 위치와 같이 제어없이 주기적으로 수집되는 데이터(Passive data)를 구분하여 Passive data의 조건에서 Active data가 발생할 조건부확률을 통해 인증 여부를 판정하였다. 이 연구의 결과에서 장소의 성격에 따라 추가적인 인증 기법을 적용하는 것이 효율적임을 알 수 있으며 이러한 적용 방법을 통해 암시적 인증 기법의 보안 수준을 더욱 강화할 수 있다는 것을 소개하고 있다. [9]

3. 순환신경망을 활용한 맥락 데이터 기반 사용자 인증 기법

본 장에서는 순환신경망을 활용한 맥락 데이터 기반 사용자 인증 기법에 대해서 설명한다. 해당 기법은 사용자가 모바일 기기를 사용하면서 수집 가능한 데이터를 집계하여 인공신경망 모델을 생성하는 것이다. 생성된 모델을 사용자의 인증 요청에 대한 응답으로 0과 1사이의 실수인 인증 점수를 산출한다. 모델을 학습하기 위한 데이터로 사용자의 위치기록, 통화기록, 앱 사용기록을 사용하였으며, 데이터의 형태는 그림2과 같다.

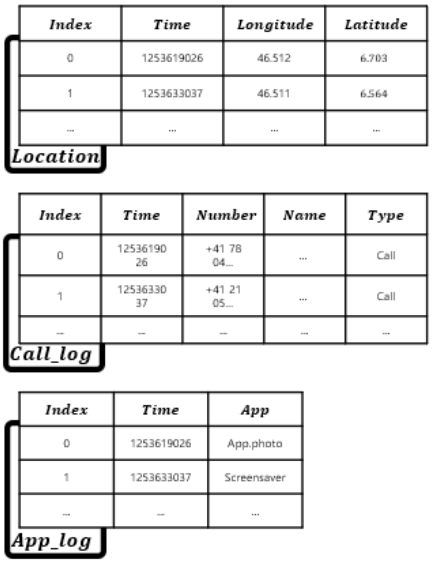


그림 2. 맥락 데이터

순환신경망을 활용한 맥락 데이터 기반 사용자 인증 기법은 그림과 같이 세 단계로 이루어져 있으며, 자세한 설명은 따라오는 각 세부 장에서 설명한다.

3.1. 집 계

그림 3. 집계 과정에 따른 데이터 변화

모델 생성의 첫 단계인 집계 단계에서는 사용자의 맥락 데이터를 수집하여 30분 단위로 위치, 대상, 애플리케이션별 빈도를 계산한다. 따라서 그림3과 같이 사용자의 모든 데이터는 30분 단위로 나뉘어진 다수의 데이터로 바뀌게 된다.

3.2. 데이터 통합

데이터 종류마다 하나의 신경망을 구성하여 여러 개의 모델을 생성할 경우, 각 데이터 사이의 연관성을 반영하지 못하기 때문에, 여러 종류의 데이터를 순환신경망 하나에서 처리하기 위해서는 각 데이터를 하나의 입력데이터로 변환할 필요가 있다. 데이터 통합 단계에서는 각 시간 구간에 맞는 데이터들을 하나의 행으로 연결시켜 입력데이터로 구성하여 각 데이터를 시간에 따른 하나의 데이터로 변경하였다.

3.3. 모델 훈련

순환신경망은 순차적인 데이터를 처리할 수 있는 신경망의 한 종류이다.[11] 따라서 순차적인 데이터의 특징을 갖은 언어처리 등에 쓰이고 있으며, 맥락 데이터 또한 순차적인 데이터이기 때문에 맥락 데이터 기반 인증 기법에 어울리는 신경망 모델이라고 볼 수 있다. 본 논문에서 사용한 순환신경망은 그림1의 Recurrent Neural Networks와 같이 구성되어 있다. 하나의 레이어를 사용한 단층신경망으로 구성되었으며, 총 5개의 노드를 사용하여 요청 시점으로부터 150분의 데이터를 통해 사용자 인증 점수를 평가하였다. 인증 점수를 0과 1사이의 실수로 계산하기 위해서 Score를 만들어내는 마지막 노드에서는 출력의 Activation Function으로 Sigmoid를 사용하였다. 모델의 학습은 3.2절에서 설명한 데이터를 입력으로 하며, 이때 데이터의 label은 사용자의 데이터일 때 1, 타인의 데이터일 때 0을 사용하였다.

4. 검 증

제안한 사용자 인증 점수 산출 기법의 효용성을 검증하기 위해서 MDC(Mobile Data Challenge) dataset을 사용한 실험을 진행하였다. MDC dataset은 Nokia Research를 통해 수집된 데이터로 200여명의 사용자에 대한 2년간의 위치, 통화, 애플리케이션 외에 Wifi, Calendar, bluetooth등 다수의 데이터를 포함하고 있는 데이터셋이다.[12] 검증 단계에서는 MDC dataset의 10명의 사용자의 데이터에 무작위 타인의 데이터를 포함시킨 학습 데이터와 다른 10명의 같은 형태의 테스트 데이터를 사용하였다.

User ID	Accuracy	
	TrainingSet	TestSet
5937	0.8370	0.8783
5939	0.6903	0.6173
5940	0.7570	0.6927
5942	0.6050	0.4047
5943	0.8969	0.8357
5949	0.6096	0.5928
5959	0.5325	0.6159
5961	0.6923	0.6196
5964	0.7606	0.7171
5966	0.6467	0.6699
Average	0.7027	0.6644

표 1. 검증결과

데이터의 전처리와 순환신경망의 학습 및 평가는 Pycharm 개발 환경에서 Python3.6, Tensorflow1.12.0, Keras 2.2.4를 사용하였다.

실험을 통한 결과는 아래의 표1과 같이 정확도의 평균이 학습 데이터에서 0.7027, 테스트 데이터에서 0.6644의 결과를 얻을 수 있었다. 또한 사용자별로 정확도의 편차가 큰 것을 볼 수 있는데, 이것은 사용자마다 수집된 맥락 데이터 사이의 시간 차이가 각기 다르고, 행동 패턴의 일정한 정도도 다르기 때문으로 생각되며, 더불어 수집된 데이터의 공백이 다수 분포하고 있다는 점이 모델의 정확도에 부정적인 영향을 주었다고 생각한다.

5. 결론

본 연구에서는 사용자의 위치, 통화기록, 애플리케이션 기록을 사용하여 순환신경망을 학습시키고, 생성된 신경망 모델을 통하여 사용자 인증을 수행하는 방법을 제안하였다. 또한 실험을 통하여 0.6644에 해당하는 정확도를 얻을 수 있었다. 비록 정확도가 매우 높은 수준은 아니지만, 해당 기법이 사용자의 제어가 불필요한 점과 본인의 인증이 실패한 경우 비밀번호 등의 다른 인증 기법을 추가적으로 사용할 수 있다는 점에서 충분한 장점이 있기 때문에 활용성이 있다고 볼 수 있다. 향후 연구를 통하여 여러 타입의 데이터를 하나로 연결하지 않으면서 하나의 신경망으로 평가할 수 있는 구조로 모델을 수정할 계획이며, 데이터의 전처리 과정에 적용할 더욱 타당한 방법에 대하여 연구할 계획이다.

6. Acknowledgement

이 논문은 정부(정보통신기술진흥센터)의 재원으로 대학ICT연구센터육성지원사업의 지원을 받아 수행된 연구임(No. IITP-2018-2017-0-01628).

7. 참고문헌

[1] L. Whitney. Smartphones to dominate PCs in Gartner forecast. CNET Business Tech News. 2010

[2] L. Rainie and J. Anderson. The Future of the Internet III. Pew Internet Project. 2008.

[3] C. Nachenberg, “A window into mobile device security: Examining the security approaches employ. *SSR*. 2011.

[4] 정기석. 국내 모바일 간편결제 활성화 방안의 관한 연구. 융합보안논문지 제 15권 제 4호 p73-p82. 2015.

[5] Feng Zhang, Aron Kondoro, Sead Muftic. Location-based Authentication and Authorization Using SmartPhone. *TrustCom.* 2012.

[6] Elaine Shi, Yuan N., Markus J. and Richard C. Implicit Authentication through Learning User Behavior. *ISC.* 2010.

[7] Markus J., Elaine Shi, Philippe G., Richard C. Implicit Authentication for Mobile Devices. *USENIX* . 2009.

[8] Hilmi G. K., Mike Just, Lynne B., David A. and Nicholas M. Data Driven Authentication: On the Effectiveness of User Behaviour Modelling with Mobile Device Sensors. *MoST.* 2014.

[9] Eiji H., Saivik Das, Shahriyar A., Jason Hong and Ian Oakley. CASA:Context-Aware Scalable Authentication. *SOUP.* 2013.

[10] Tomas M., Martin K. Recurrent Neural Network based language model. *INTERSPEECH.* 2010.

[11] Rumelhart D., Hintim G., Williams R. Learning representations by back-propagating error. Nature, 323, 533-536. 1986.

[12] Dataset distribution portal. “Mobile Data Challenge(MDC) Dataset”. https://www.idiap.ch/dataset/mdc. 2012. (2018-10-28 방문).

MDA 기반 태양광 에너지 모니터링 시스템의 유지보수를 위한 비용 감소 방안 사례

장우성 O1, 김영철 2

홍익대학교 소프트웨어공학연구실 1,2

{jang1, bob2}@selab.hongik.ac.kr

A Case study of Cost Reduction for Maintenance of Photovoltaic Energy Monitoring System based on Model Driven Architecture (MDA)

Woo Sung Jang O1, R. Young Chul Kim2

SE Lab, Dept. of Software & Communications Engineering, Hongik University1,2

요 약

현재 국내에는 많은 태양광 설비 업체들이 다양한 회사의 인버터 장비를 사용하여 발전 설비를 구축하고 있으며, 인버터는 새로운 종류가 지속적으로 추가되는 실정이다. 이종 인버터 데이터의 쉬운 관리 및 유지보수를 위해 M-PVMS가 개발되었다. 하지만 메타모델과 변환 규칙이 모두 자바로 구현되어 있기 때문에, 시스템을 유지보수 시 코드 수정 및 컴파일을 위해 많은 비용이 발생한다. 본 논문은 기존 M-PVMS의 유지보수성을 높이기 위해 XMI, ATL, ECore를 이용하여 기존 메타모델과 메타모델 변환 엔진을 개선하는 방법을 제안한다. 유지보수 과정에서의 시스템 복잡도 증가 가능성을 예방하고, 시스템 유지보수 환경 구축에 소비되는 비용을 줄일 수 있다.

1. 서 론

태양광 에너지 발전 사업은 대표적인 신재생 에너지 사업으로써[1,2], 다양한 패널과 인버터 제품이 출시가 되고 있다. 태양광 에너지 모니터링 시스템[3]에 새로운 인버터를 추가한다면, 새로운 프로토콜 추가를 위한 시스템 유지보수가 필요하다. M-PVMS는 메타모델을 이용하여 인버터 프로토콜 데이터를 변환 및 통합하고, 태양광 발전량 데이터를 관리 및 모니터링 한다. 메타모델과 변환룰의 추가를 통해 이종 인버터 추가가 가능하다[4]. 하지만 자바를 기반으로 제작되었기 때문에, 모델과 룰의 추가를 위해 자바 소스코드 수정이 필요하다. 그리고 내부적으로 다양한 라이브러리가 사용되기 때문에 해당 라이브러리들이 모두 설치된 개발 환경에서 컴파일이 가능하다. 만약 개발 환경이 없는 상태에서 시스템의 유지보수를 수행한다면 이클립스 도구와 추가적인 라이브러리 설치 작업이 필요하다. 또한 시스템 구조의 이해를 위해 기존 자바 코드들을 분석해야 한다.

본 논문은 이러한 문제의 해결을 위해 메타모델과 변환룰의 개선 방안을 제안한다. 메타모델을 ECore[7] 기반의 XMI(XML Metadata Interchange)[7] 코드로 표현한다. 모델 변환룰을 ATL(Atlas Transformation Language)[8] 기반 스크립트 언어로 표현한다. 기존 인버터 메타모델을 개선한다. 마지막으로, 개선된 요소들이 포함된 메타모델 자동 변환 환경을 제안한다. 범용적인 스크립트 언어와 다이어그램을 사용함으로써 변환 알고리즘의 이해도를 높일 수 있다. 그리고 스크립트 코드 파일만 추가함으로써, 새로운 메타모델과 변환룰의 추가가 가능하다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로써, M-PVMS를 언급한다. 3장은 이종 태양광 메타모델 자동 변환 환경을 설계한다. 4장은 설계한 변환 환경을 구현한다. 마지막으로, 결론 및 향후 연구를 언급한다.

2. M-PVMS(Metamodel based Photovoltaic Monitoring System)[4,5]

M-PVMS는 Model to Model 변환 구조[6]를 이용하여 이종의 인버터 데이터를 쉽게 통합하고, 추후 추가되는 이종 인버터 패킷을 기존 알고리즘의 수정 없이 추가할 수 있도록 지원하는 태양광 에너지 모니터링 시스템이다.

M-PVMS의 모델 변환 과정은 그림 1과 같다. 인버터가 UART 통신을 사용하여 M-PVMS Client에게 발전량 데이터를 전송한다. M-PVMS Client는 Data Collector를 사용하여 전송받은 데이터를 분석 후 내부적으로 인버터 모델로 만들고, 자바 기반의 변환룰을 통해 통합 태양광 에너지 모델로 변환한다. 변환이 완료된 모델은 XMI 코드로 표현된다. M-PVMS Client는 TCP/IP 통신을 사용하여 M-PVMS Server에게 XMI 코드를 전송한다. M-PVMS Server는 전송 받은 XMI 코드를 분석하여 데이터베이스에 저장한다.

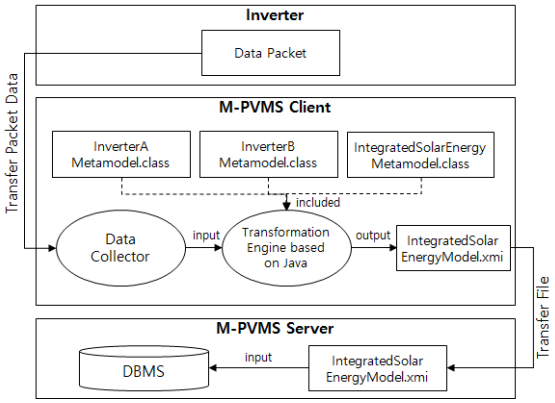


그림 1 M-PVMS의 자동 모델 변환 과정

3. 이종 태양광 메타모델 자동 변환 환경 설계

3.1 개선된 메타모델 자동 변환 환경 설계

개선된 메타모델 자동 변환 과정은 그림 2와 같다. 인버터는 M-PVMS Client에게 발전량 데이터를 전송한다. DataColletRule.xmi 파일은 이종 인버터들의 데이터 요청 및 응답 스키마이다. M-PVMS Client의 Data Collector는 DataColletRule.xmi 파일을 읽어 인버터의 패킷을 해석하고, 인버터 모델을 생성하여 xmi 파일로 저장한다. 인버터 모델과 통합 태양광 에너지 모델은 ecore 기반의 다이어그램으로 작성된다. 다이어그램은 내부적으로 xmi 코드로 표현된다. 모델 간의 변환룰은 ATL을 사용하여 TranslateRule.atl 파일 내에 저장된다. 변환 엔진은 TranslateRule.atl을 실행하고, TranslateRule.atl은 ecore로 표현된 메타모델을 참조하여 모델 변환을 수행한다. 변환이 완료된 모델은 xmi 파일로 저장된다. M-PVMS Client는 저장된 xmi 파일을 M-PVMS Server에게 전송한다. M-PVMS Sever는 전송받은 파일을 해석하여 데이터베이스에 저장한다.

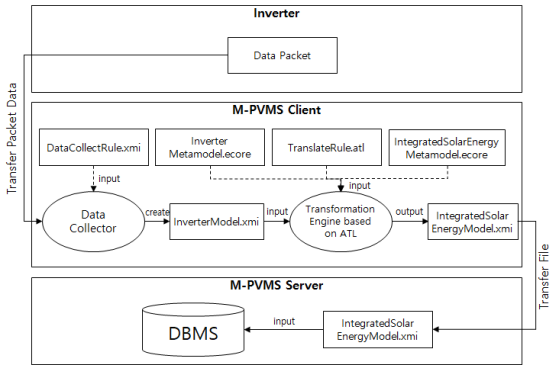


그림 2 개선된 M-PVMS의 자동 모델 변환 과정

3.2 인버터 메타모델 구조 개선

본 장은 개선된 환경에 적합한 인버터 메타모델을 제안한다. 인버터 패킷은 인버터 제조사 별로 패킷의 길이와 종류가 다르다. 본 논문은 패킷들의 공통된 구조를 파악하여, 그림 3과 같이 메타모델을 설계한다. 인버터 패킷의 공간은 크게 Header, Data, Tail로 나뉜다. Header는 패킷의 타입이나 길이와 같은 정보가 저장된다. Data는 발전량, 알람, 또는 센서 데이터 등이 저장된다. Tail은 Checksum이나 기타 데이터가 저장된다.

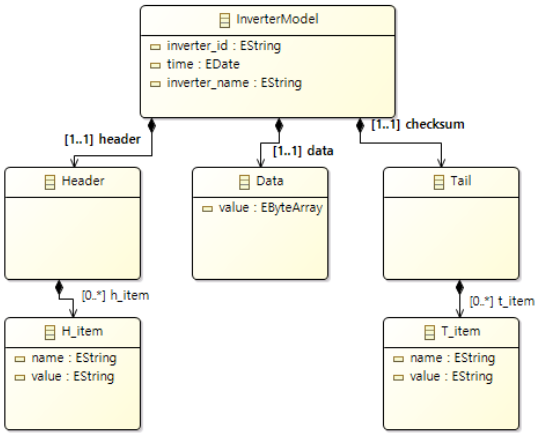


그림 3 개선된 인버터 메타모델

3.3 ATL 기반의 메타모델 자동 변환 엔진 설계

그림 4는 개선된 메타모델 자동 변환 엔진의 구조를 나타낸다. 인버터 모델은 소스 모델의 역할을 수행하고, 통합 태양광 에너지 모델은 타깃 모델의 역할을 수행한다. 변환 언어는 ATL을 사용한다. ATL 기반 변환 알고리즘은 *.atl 파일 내에 저장되며 변환 엔진은 atl 파일을 실행하여 변환을 수행한다.

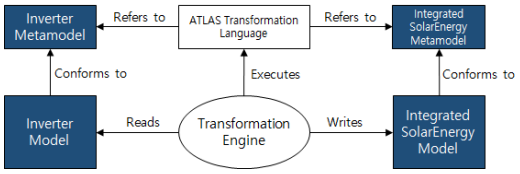


그림 4 메타모델 자동 변환 엔진

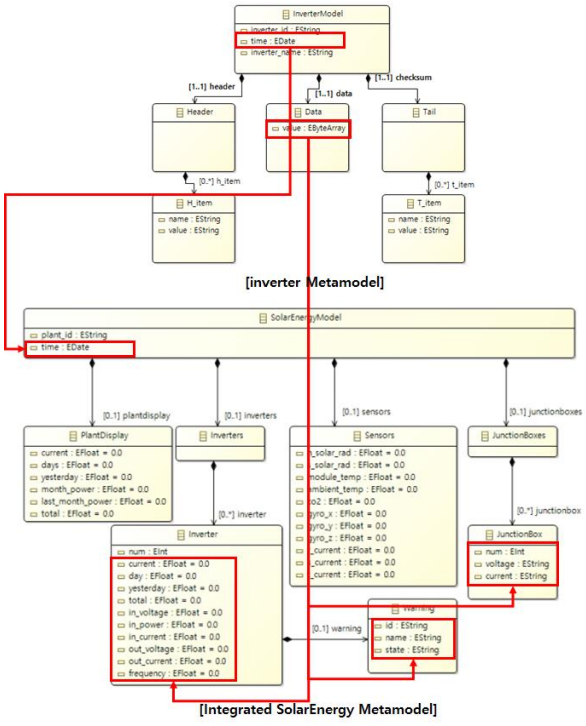


그림 5 메타모델 변환 흐름

그림 5는 인버터 모델의 변환 흐름을 나타낸다. 상단의 메타모델은 인버터 메타모델이고, 하단의 메타모델은 통합 태양광 에너지 메타모델[5]이다. 화살표는 실제 변환되는 부분을 나타낸다.

InverterModel 클래스의 time은 SolarEnergyModel 클래스의 time으로 변환된다. H-item 클래스에는 인버터 모델의 타입이 저장되는데, 현재 인버터 모델이 발전량 데이터를 저장한 모델이라면, ATL룰을 이용하여 Data 클래스의 value를 Inverter 클래스의 발전량 데이터로 변환한다. 만약 현재 인버터 모델이 알람 데이터를 저장한 모델이라면, Data 클래스의 value를 warning 클래스의 알람 데이터로 변환한다. 만약 인버터 모델이 계통 데이터를 저장한 모델이라면, Data 클래스의 value를 JunctionBox의 계통전압 데이터로 변환한다.

4. 이종 태양광 메타모델 자동 변환 환경 적용

인버터 메타모델을 이용하여 생성된 인버터 XMI 코드는 아래의 그림 6과 같다.

```
<?xml version="1.0" encoding="ASCII"?>
<inverter:InverterModel xmi:version="2.0"
xmlns:xmi=http://www.omg.org/XMI
xmlns:inverter="http://selab.hongik.ac.kr/inverter"
inverter_id="1" inverter_name="HS" time="2017-0807T20:50:10.000+0900">
  <header>
    <h_item name="sop" value="^"/>
    <h_item name="code" value="D"/>
    <h_item name="address" value="4"/>
    <h_item name="length" value="19"/>
    <h_item name="id" value="000"/>
  </header>
  <data value="001,0078,0000100,31"/>
  <tail>
    <t_item name="checksum" value="24"/>
  </tail>
</inverter:InverterModel>
```

그림 6 인버터 모델의 XMI 코드

ATL 기반 변환 엔진을 통해 생성된 통합 태양광 에너지 모델의 XMI 코드는 아래의 그림 7과 같다.

```
<?xml version="1.0" encoding="ASCII"?>
<hsSolarEnergy:SolarEnergyModel xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:hsSolarEnergy="http://selab.hongik.ac.kr/hsSolarEnergy"
plant_id="0410001" time="2017-08-07T20:50:10.000+0900">
  <plantdisplay days="128510.0" yesterday="125940.0"
  month_power="886656.0" last_month_power="2434944.0"
  total="5.7089608E7"/>
  <inverters>
    <inverter num="1" day="128510.0" yesterday="125940.0"
    total="5.7089608E7" in_voltage="3.0" out_voltage="229.33"
    frequency="60.0">
      <warning id="pv_2;inverter_12;"
      name="&#xc800;&#xc804;&#xc555;;MC_A;"
      state="&#xd0dc;&#xc591;&#xc804;&#xc9c0;
      &#xc800;&#xc804;&#xc555;;;"/>
    </inverter>
  </inverters>
  <sensors/>
  <junctionboxes>
    <junctionbox num="1"
    voltage="0;0;5;0;0;0;0;0;408;409;410;411;412;413;414;415;416;417;418;4
    19;420;421;422;423;424;425;426;427;428;429;430;431;"
    current="1.1;1.2;1.1;1.6;1.4;1.2;0.0;0.0;20.8;20.9;21.0;21.1;21.2;21.3
    ;21.4;21.5;21.6;21.7;21.8;21.9;22.0;22.1;22.2;22.3;22.4;22.5;22.6;22.7
    ;22.8;22.9;23.0;23.1;"/>
  </junctionboxes>
</hsSolarEnergy:SolarEnergyModel>
```

그림 7 통합 태양광 에너지 모델의 XMI 코드

결과적으로, 기존 M-PVMS는 시스템 내 메타모델과 변환룰의 추가 및 수정을 위해 자바 소스코드를 직접 수정해야 했지만, 제안한 방법은 인버터 모델의 XMI 코드와 ATL 기반 변환 법칙을 파일 형태로 추가하여 시스템의 유지보수가 가능하다.

5. 결 론

본 논문은 기존 메타모델 기반 태양광 에너지 모니터링 시스템의 유지보수성을 향상시키기 위한 방법을 제안한다. 자바 언어로 구현된 변환 엔진을 ECore, XMI, ATL로 재구성하고, 인버터의 메타모델을 개선된 구조에 맞게 재설계하였다. 모델 변환룰과 메타모델을 외부파일로 저장하도록 구성하여 기존 시스템의 재컴파일 없이 새로운 메타모델이 추가될 수 있도록 하였다. 새로운 이종 인버터가 추가된다면, 데이터 요청/응답 패킷의 스키마를 XMI 파일 내에 추가하고, ATL 변환룰이 저장된 ATL 파일을 수정함으로써, 쉽게 메타모델을 추가 및 관리할 수 있다. ECore, ATL 개발 환경은 이클립스 플러그인으로 제공되기 때문에 메타모델 및 변환룰의 쉬운 제작이 가능하다.

참고 문헌

[1] “미래를 준비하는 각 지역의 에너지 정책”, 대한지방행정공제회.

[2] 정용대, 남유진, “태양광열·지열 하이브리드 시스템의 건물형태 및 용도별 도입타당성 분석”, 2016 대한건축학회 추계학술발표대회논문집, 제36권, 제2호, 609-610, 2016.10.

[3] 이세현, 신상욱, 이현영, 권기태, 나종혁, 최현석, “태양광발전시스템의 옥외 실증 모니터링 기술동향”, 한국신재생에너지학회 2017년도 춘계학술대회[초록집], 183-183, 2017.05.

[4] Hyun Seung Son, R. Young Chul Kim, “Modeling a Photovoltaic Monitoring System based on Maintenance Perspective for New & Renewable Energy”, The 2nd International Joint Conference on Convergence 2016, AACL 07, 144-147, 2016.

[5] Woo Sung Jang, Hyun Seung Son, Chae Yun Seo, R. Young Chul Kim, “Metamodel based Photovoltaic Monitoring System for Heterogeneous Renewable Energies”, 2017 International Symposium for Advanced Computing and Information Technology.

[6] K. Czarnecki, S. Helsen, “Feature-based survey of model transformation approaches”, IBM Systems Journal. Vol.45, No.3, 621-645, 2006.

[7] F. Budinsky, E. Merks, D. Steinberg, EMF: Eclipse Modeling Framework, Addison-Wesley, 2009.

[8] Wikipedis, ATL, http://en.wikipedia.org/wiki/ATLAS_Transformation_Language

위해도 분석 결과의 효과적인 확인을 위한 추적성 기반 위해도 모델

정세진O, 유준범
건국대학교 컴퓨터정보통신공학과
{jsjj0728, jbyoo}@konkuk.ac.kr

Hazard model for efficiency of hazard analysis based on traceability relations

Sejin JungO, Junbeom Yoo
Division of computer science and engineering, Konkuk university

요 약

위해도 분석은 시스템 또는 소프트웨어의 사고가 될 수 있는 잠재적인 위험 상태를 분석하는 것으로 안전성이 중요한 시스템에 필수적으로 적용되고 있다. 현대의 시스템은 과거에 비해 그 복잡도가 급격하게 증가하고 있어 위해도 분석이 쉽지 않다. 따라서 현재 전체 시스템에 하나의 기법을 활용한 분석 보다는 개발 생명주기, 구조에 따라 시스템/서브시스템/컴포넌트 수준에서 여러 기법들이 사용되는 것이 일반적이다. 이 때 각각 적용한 위해도 분석은 원인-결과의 연결이나 상호 작용에 의한 관계를 가질 수 있다. 하지만 현재 이런 관계에 대해 추적성 분석 수준의 연구만 진행되고 있어 전체적인 확인이 어렵다는 문제점이 있다. 본 논문에서는 여러 위해도 분석 결과의 효과적인 확인을 할 수 있는 추적성 기반 위해도 모델을 제안한다. 위해도 모델은 분석 결과의 모델링뿐만 아니라 각 요소들의 관계를 정의해 포함하고 있다.

1. 서 론

안전 필수 시스템 (safety-critical system)은 사고로 인한 결과가 인명피해, 환경오염과 같이 심각한 문제가 되는 중대한 시스템을 의미한다. 따라서 안전 필수 시스템은 사용 전 안전성 분석 등을 통해 그 안전성에 대해 입증해야 한다[1].

위해도 분석 (hazard analysis)은 시스템/소프트웨어의 사고가 될 수 있는 잠재적인 위험 (hazard)을 분석하는 기법으로 안전성을 확보하기 위해 위험을 확인하고 제거하는 목적으로 사용하는 방법이다. 기능 안전성 표준에서는 위해도 분석을 통해 시스템의 hazard, risk를 확인하고 안전 요구사항 (safety requirement)을 할당하는 프로세스를 가지고 있어 위해도 분석은 안전 필수 시스템 개발에 필수적으로 수행되어야 한다. 위해도 분석에는 FMEA (Failure Mode & Effect Analysis), FTA (Fault Tree Analysis)와 같이 전통적으로 많이 사용되던 기법에서 STPA (System Theoretic Process Analysis), ECFA (Event Causal Factor Analysis) 등 수많은 기법들이 개발되어 적용되고 있다[2][3].

현대의 시스템은 과거에 비해 그 기능이 다양해지고 복잡도가 급격하게 증가하는 등 위해도 분석이 어려워지고 있다. 따라서 하나의 기법을 통해 전체를 분석하기 보다는 생명주기, 계층적 구조 등에 따라 다양한 기법들이 적용되는 것이 일반적이다. 시스템 이론 (system theory)에 따르면 이런 시스템은 독립적인 컴포넌트들의 단순한 집합체가 아닌 각각의 컴포넌트들이 유기적으로 상호작용하는 관계이며, 각각 적용한 위해도 분석 결과들도 효과적으로 각각의 관계를 확인 해 볼 필요가 있다. 이 때 각 위해도 분석의 결과들은 다양한 연결 고리들을 가질 수 있으며 추적 관계 분석을 통해 연결성, 관계 등을 효과적으로 확인할 수 있다[6].

하지만 현재는 주로 독립적으로 위해도 분석을 적용해 결과를 도출하거나, 계층적 구조에 대해 원인-결과의 분석 가능성에 대해서만 언급하고 있다[4]. 또는 전체 안전성 분석 측면에서 추적성을 확인하는 정도에서 그치고 있는 등[5] 연계해서 확인하는 방법이 부족하다.

이에 따라 본 논문에서는 다양한 분석 기법을 활용한 시스템/소프트웨어의 위해도 분석 결과의 상관관계 분석을 통한 다각적인 위험, 효과, 과정 (consequence) 확인 및 모델링을 위해 추적성을 기반으로 한 위해도 모델을 제안한다. 위해도 모델은 메타모델을 활용한 분석 결과의 모델링 및 각 요소들의 관계를 정의해 표현하였다. 이를 통해 여러 종류의 위해도 분석 기법을 사용한 분석에 도움이 될 수 있다.

2. 배경지식

2.1. 위해도 분석 (hazard analysis)

위해도 분석은 시스템/소프트웨어가 사고가 될 수 있는 잠재적인 위험 (hazard)을 식별하는 기법으로 현재 다양한 기법들이 사용되고 있다. 현재 주로 사용되는 기법으로는 FMEA, HAZOP (Hazard & Operability)), FTA가 있으며, STPA도 최근 많이 연구되고 있는 기법 중 하나이다.

FMEA는 subsystem, component, function의 잠재적인 고장 모드 (failure mode)가 시스템에 미치는 영향을 상위 수준으로 분석하는 귀납적 분석 방법이다[2]. FMEA를 활용해 하위의

작은 모듈/컴포넌트의 문제가 전체 구조에 어떤 영향을 미치는지 분석하는데 적합한 방법으로 의도하지 않은 상황을 확인 할 수 있다. FMEA는 주로 아래 <그림 1>과 같은 worksheet table을 활용해 분석을 수행하며, 대상의 잠재적인 failure mode 확인으로부터 시작해 그 영향 및 risk 를 분석하는 방법이다.

FMEA를 수행하기 위해서는 시스템의 functional 또는 structural decomposition이 중요하며 worksheet에서 item을 결정하는 부분이 된다. 또한 failure mode를 올바르게 확인하는 것이 중요하며 이 부분에 대한 연구들 또한 존재한다.

System:				Subsystem:					
Item	Failure mode	rate	Cause	Immediate effect	System effect	Detection	Hazard	Risk	Recommend

그림 1 FMEA worksheet 예제

STPA는 시스템 이론에 기반해 시스템의 안전을 고장의 문제가 아닌 제어의 문제로 분석하는 기법이다. 분석은 각각 컴포넌트의 고장 모드가 상위 수준에 미치는 영향을 분석하는 것이 아닌 각 컴포넌트의 컨트롤 명령 (control action)을 통해 상호작용을 모델링한 control structure를 작성하고, control structure로부터 위험이 발생할 수 있는 명령 및 원인을 분석하는 기법이다. 시스템을 STAMP (System Theoretic Accident Model and Process)기반의 모델로 작성하고 위험을 유발할 수 있는 제어 (Unsafe control action)도출 및 원인 확인으로 진행된다.

이외에도 FMEA와 유사한 worksheet을 이용하는 HAZOP 기법과 FTA와 유사한 이진 트리 형태로 분석을 수행하는 ETA (Event Tree Analysis), 문제가 발생하는 이벤트 연결 및 원인을 분석하는 ECFA, 안전성 분석의 구조적인 논증 구조를 제공하는 safety case[7] 등 다양한 기법들을 사용할 수 있다.

2.2 관련연구

위 절에서 설명한 바와 같이 위해도 분석을 위해 사용할 수 있는 기법들은 수많은 종류들이 있다. 기법에 따라 고장으로부터 그 결과를 분석하거나, 결과로부터 그 원인을 구성하는 기법 등 서로 다른 방법, 목적을 가지고 있다. 이 때 이러한 기법들을 통합해서 분석하고자 하는 노력들이 존재한다. 가장 자주 나타나는 방법은 FMEA와 FTA가 서로 반대의 분석을 목적으로 하는 과정에 착안하여 FMEA의 결과를 FTA를 통해 원인을 분석하거나, FTA의 결과를 FMEA를 통해 consequence를 재 분석하는 방법에 대한 연구들이 있다[8]. 그 외에도 STPA 분석 중 UCA의 원인 분석을 FTA를 통해 수행하는 통합에 대한 연구 또한 존재한다.

[5] 에서는 시스템 개발 시 안전성 분석 전 과정에 걸쳐 안전성 분석 결과, 개발 산출물, 안전 요구사항 및 위험 경감에 대한 추적성 분석에 대해 제안하였다. 하지만 해당 논문에서는 안전성 분석 전 과정에 더 초점을 맞추고 있어 위해도 분석의 추적성 분석에 대한 언급이 부족하다.

3. Hazard model

2.1 절에서 설명한 바와 같이 위해도 분석에 적용될 수 있는 분석 기법들은 다양한 기법들이 있으며 해당 기법들은 적용 방법, 목적 등에 조금씩 차이가 있다. 따라서 추적성 기반의 hazard model을 위해서는 각 기법에 대해 확인할 필요가 있다. 본 논문에서는 이를 모델링 하기 위해 각 위해도 분석 기법의 과정에 따라 각 기법의 1차 분류를 수행하고 각 분류를 기반으로 연결 관계 분석 및 메타모델을 개발하였다. 분류를 위해 사용한 기법은 특수 목적만을 위해 사용되는 방법을 제외하고 선정하였다.

3.1 기법의 추상화된 모델

위해도 분석에 사용되는 다양한 기법들은 서로 간의 차이점과 공통점을 가지고 분류할 수 있다. 본 논문에서는 기법들의 분석 방법의 공통점에 따라 이를 크게 5 +1 가지로 분류하였다. 본 논문에서 수행한 분류는 각 기법들의 특징을 추상화해 요소를 추출할 수 있는 기준으로 수행하였으며 이를 바탕으로 추상화된 다이어그램을 개발하였다. <표 1>은 본 논문에서 제안하는 위해도 분석 기법의 분류로 분류는 tree를 사용하는 기법들, worksheet을 사용하는 기법들, 다이어그램을 사용하는 두 분류의 기법, 그 외 시나리오 분석 및 safety case로 구성한다.

Tree 다이어그램을 사용하는 기법은 FTA, ETA와 같이 fault tree/event tree를 작성하고 이를 바탕으로 분석을 수행하는 기법들이 해당한다. Worksheet 기반의 분류는 FMEA, HAZOP과 같이 worksheet table을 이용하는 기법들이 속하며 그 외 PHA (Preliminary Hazard Analysis), FHA (Fault Hazard Analysis), SHA (System Hazard Analysis) 등이 있다. 다이어그램 분류는 fault tree외의 다이어그램을 사용하는 기법들이 속하며 위에서 설명한 STPA와 이벤트 나열을 통한 분석 기법인 ECFA/STEP 등이 해당된다.

표 1 적용 방법에 따른 위해도 분석 기법의 분류

Classification	Description	Technique
Tree diagram	Tree 형태의 다이어그램을 이용하는 분석 기법	FTA, ETA, CCA
Worksheet	Worksheet table을 이용하는 분석 기법	FMEA, HAZOP, PHA, Etc.
Diagram	STPA	-
	ECFA/STEP: Event chart를 이용하는 기법	-
Other	그 외 시나리오 분석	Checklist, What-if, Etc.
Safety case	Safety case	-

다음으로 기타 시나리오 분석 분류는 checklist, what-if analysis, scenario analysis와 같이 직접적인 위해도 분석도 가능하지만, 질문을 통해 해당 시스템이 어떤 위험을 내포하고 있는지 도움을 주는 리스트 기반의 기법으로 구성하였다. 마지막으로 safety case는 다른 기법들과 달리 직접적인 분석을 수행하는 기법은 아니지만 각 결과를 활용한 논증 구조를 작성해 safety demonstration, argument 등을 제공하는 기법이므로 분류에 포함하였다.

다음으로는 각 분류에 해당하는 기법들의 시작점 분석을 통해 분석 요소들을 추상화하였다. 시작점은 위해도 분석의 시작이 되는 지점을 정의한 것으로 FMEA는 item/component의 고장 모드로부터 그 결과를 분석하는 기법이기 때문이 처음(cause) 부분이 시작점이 된다. 반면 FTA는 failure/accident로 작성되는 top-event로부터 그 원인을 논리적으로 분석하는 기법이기 때문에 결과(result) 부분이 시작점이 될 수 있다.

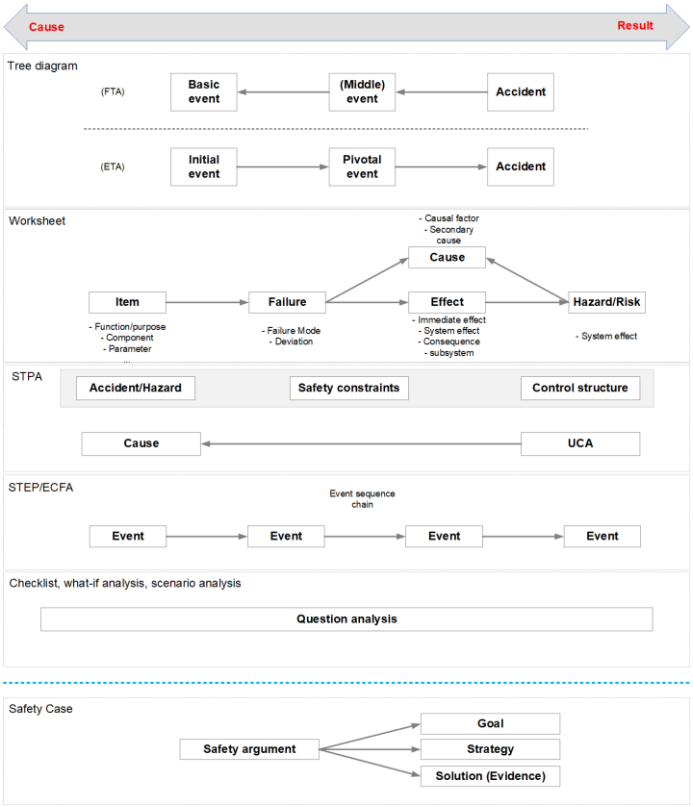


그림 2 기법의 분류에 따른 추상화 다이어그램

<그림 2>는 각 기법별로 시작점 분석의 결과로 도출된 위해도 분석의 방향에 따라 필수적인 요소들을 추출해 다이어그램으로 작성한 그림이다. Tree diagram에 속하는 FTA의 경우 accident (top event)부터 중간에 작성되는 event및 basic event로 구성되며, ETA의 경우 FTA와는 달리 초기 사건 (initial event)로부터 accident가 되는 과정의 이벤트로 구성된다. Worksheet 분류의 기법들의 경우 기본적으로 item으로부터 failure, effect, hazard/risk를 분석하는 과정으로 진행되며, 각 기법에 따라 요소들이 변화될 수 있음을 포함하고 있다.

3.2 기법의 연결 관계 분석

추적성 분석 (traceability analysis)는 소프트웨어 개발 시 요구사항과 다른 명세간의 관계를 정의하는 것으로 요구사항 명세가 디자인, 구현 단계를 거쳐 빠짐없이 구현되었는지 확인하거나, 관계 정의를 통해 요구사항의 변경을 손쉽게 반영할 수 있도록 하는 분석이다. 본 논문에서는 추적성 분석의 개념을 위해도 분석의 각 기법별로 도출되는 요소들 간에 item/component를 기준으로 추적 관계를 분석하는 것을 추적성 기반의 분석이라 한다.

추적 관계는 <그림 2>에 나타난 각 요소들을 기준으로 생각해 볼 수 있으며 본 논문에서는 이를 2 종류로 분석하였다. 첫 번째는 기본 사용 관계 (basic usage relation)으로 기존 연구에서 FMEA와 FTA를 보완관계로 사용하는 것과 같은 추적 관계이다. 다음으로는 기초 추적 관계 (basic trace relation)로 위해도 분석 결과 별 도출되는 각 요소들에 대해 item/component를 기준으로 추적 관계를 분석하는 추적성이다. Worksheet, STPA와 같이 item 항목이 명확하게 정의되는 경우는 해당 항목을 사용하며 FTA의 basic event와 같이 명확하게 정의되지 않는 경우는 간접적으로 드러나는 정보를 활용해 연결을 수행할 수 있다. 추가적으로 드러나는 이론적 의미의 추적성도 일부 포함하고 있다. 서로 다른 기법을 활용한 위해도 분석에서 같은 failure, 같은 hazard, 같은 과정의 경우를 연결하는 추적 관계이다. <표 2>는 해당 추적 관계의 일부에 대한 표이다.

표 2 각 기법의 추적 연결관계 (일부)

Relation between (from – to)	Description
(1) Basic event – (2) Failure	Failure equivalent trace
(1) Basic event – (2) Item	Component related trace
(1) Accident – (2) Hazard/Risk	Accident equivalent trace
(2) Item – (3) control structure	Basic component trace
(2) Effect/Hazard – (3) UCA/state variable	Failure relation
(2) Failure/effect/hazard – (4) Event	Failure event equivalent contents traceability
(4) Event – (5) question	Question extraction traceability
....	
2 (Hazard/Risk) -> 1 (Accident)	1 (Accident)와 연결해서 2 (Hazard/Risk) 의 원인 분석
2 (Failure) -> 1 (Accident)	1 (Accident)와 연결해서 2 (Failure) 의 원인 분석
2 (Failure) -> 4 (Event)	4 (Event)의 initial 에 2 (Failure)의 입력 으로 분석 가능
2 (Hazard/Risk) – 3 (Accident/Hazard)	2 (Hazard/Risk) can help to identify the hazard for 3 (Accident/Hazard)
....	

3.3 Hazard model의 메타모델

위해도 분석 기법들의 추상화된 다이어그램과 그 연결 관계를 모델링 하기 위한 메타모델은 다음 <그림 3>와 같다. <그림 3>는 3.1절의 다이어그램에 나타난 각 요소를 바탕으로 위해도 분석의 결과 및 추적 관계를 모델링 하도록 개발된 메타모델이다. 메타모델은 기본 분석에 따른 프로세스 관계부터 추적 관계까지 모두 표현하고 있다.

<그림 3>의 (a)는 위해도 분석 결과를 (b)는 <그림 2>에서 화살표로 표현된 요소들 간의 분석 과정을, (c)는 각 요소들의 추적 관계를 모델링 하도록 개발되었다. <그림 3>의 메타모델을 활용해 위해도 분석의 결과를 표현하고, 추적

관계에 대해 용이하게 작성할 수 있다.

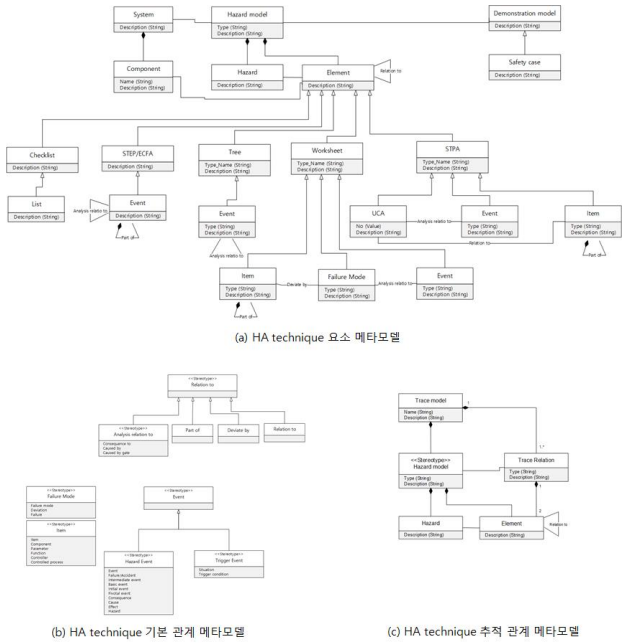


그림 3 Hazard model을 위한 메타모델

4. 사례연구

본 논문에서는 hazard model을 이용한 위해도 분석 결과의 표현을 위해 돌발상황 검지시스템 (AIDS)을 이용하여 사례 연구를 수행하였다. 현재는 제공된 정보의 한계로 인해 상위 레벨에서의 FMEA와 STPA만 적용되었다. 시스템은 교통장애를 유발하는 돌발상황을 검지하기 위한 시스템으로 시스템의 구성은 정보 수집, 처리 및 판단, 운영관리, 알림으로 구성되어 있다. <그림 4>는 돌발상황 검지시스템에 대한 FMEA와 STPA의 결과 중 일부를 메타모델을 활용해 표현한 그림이다. 아래 그림과 같이 시스템의 잠재적인 hazard에 대해 위해도 분석 요소들을 작성하는데 활용할 수 있다.

<그림 4>에 나타난 FMEA-STPA요소의 추적성은 다음과 같다. 특히 <표 3>에서 failure mode - cause 간의 추적성은 서로 다른 기법을 통해 다른 item끼리도 hazard가 연관된 추적성을 확인할 수 있음을 볼 수 있다. 이는 <표 2>에 정의한 관계 중에서 찾아볼 수 있다.

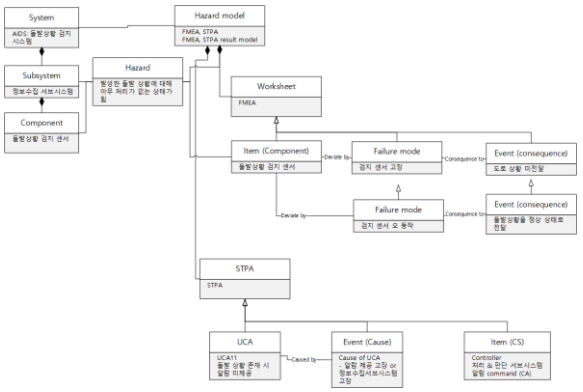


그림 4 FMEA, STPA 결과 일부에 대한 모델 표현

표 3 <그림 4>에 나타난 추적성 비교

Element		Description
FMEA	STPA	
Hazard	UCA	Same contents
Failure mode	Cause (event)	Same item traceability
Item	Cause (Event)	Implicit item traceability

5. 결론 및 향후 연구

본 논문에서는 위해도 분석 결과의 효과적인 확인 및 분석을 위해 위해도 모델을 제안하였다. 위해도 모델은 다양한 위해도 분석 기법들을 모델링하고 그 관계를 제시하였다. 이를 위해 본 논문에서는 다양한 위해도 분석 기법들을 방법, 범위에 맞게 추상화하고 추적 관계를 제시하였다. 이를 이용해 추적 관계에 대한 표현 및 작성을 용이하게 할 수 있어 위해도 파악에 도움이 될 것으로 기대한다. 향후 기법의 연결 관계에서 support 개념의 보완적 관계와 시스템 계층에 대한 고려를 포함한 모델에 대해 연구를 진행할 계획이다.

Acknowledgement

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업의 결과입니다. (NRF-2017R1D1A1B03030065)

참고 문헌

[1] Nancy G. Leveson, Jorge Diaz-Herrera, “Safeware: system safety and computers,” Addison-Wesley Reading, 1995.

[2] Clifton A. Ericson, “Hazard Analysis Techniques for System Safety”, John Wiley & Sons, 2005

[3] Nancy G. Leveson, “Engineering a Safer World: Systems Thinking Applied to Safety”, MIT Press, 2012

[4] EPRI, “Hazard analysis methods for digital instrumentation and control systems”, EPRI, 2013.

[5] Jang-Soo Lee, Vikash Katta, Eun-Kyoung Jee, Christian Raspotnig, “Means-ends and whole-part traceability analysis of safety requirements,” Journal of Systems and Software, Vol.83, No.9, pp.1612-1621, 2010.

[6] Vikash Katta, Tor St. Ihane, “A conceptual model of traceability for safety system,” 2nd International Conference on Complex Systems Design & Management, 2011.

[7] Office for Nuclear Regulation, “The purpose, scope, and content of safety cases”, An agency of HSE

[8] Fernando G Nicodemos, Carlos HN Lahoz, Martha AD and Saotome Abdala, Osamu, “Using Combined SFTA and SFMECA Techniques for Space Critical Software,” Proceedings of the 5th IAASS Conference A Safer Space for Safer World, 2012.

병행성 결함 탐지 도구에 대한 실험적 연구

박창선, 박지현, 채현아 ○, 최병주*
이화여자대학교 컴퓨터공학과
{ddangddon, pola0527, coha0112}@ewhain.net, bjchoi@ewha.ac.kr

Experimental study of concurrency fault detection tools

Changsun Park, Jihyun Park, Hyeona Chae○, Byoungju Choi*
Department of Computer Engineering, Ewha Womans University

요 약

멀티 코어 프로세서의 출현으로 소프트웨어 결함에서 병행성 결함이 차지하는 비중이 증가하고 있다. 병행성 결함이란 스레드 혹은 프로세스 간의 경쟁 조건으로 인해 발생하는 결함이다. 병행성 결함 해결을 위해 다양한 결함 탐지 방안이 연구되고 있다. 대부분의 결함 탐지 관련 연구에서는 제안하는 탐지 방안의 효과를 확인하기 위해 다양한 실험을 수행한다. 그러나 병행성 결함의 비결정적인 특성으로 인해 경험적 사례 분석이 많이 이루어지고 있다. 이에 본 연구에서는 병행성 결함 탐지 방안을 위한 실험적 연구 방법을 제안하고, 실제 수행한 사례를 보이고자 한다.

1. 서 론

이전의 단일 코어 CPU 환경에서는 멀티 스레드나 멀티 프로세스라 하더라도 진정한 동시성이 없었다. 그러나 최근 멀티 코어 CPU가 일반화되면서 동시에 공유 자원에 접근하거나 스레드의 실행 순서가 뒤바뀌는 것과 같은 병행성 결함의 발생 빈도가 높아지고 있다. 순서 위반, 원자성 위반, 교착 상태와 같은 병행성 결함은 작게는 프로그램을 멈추게 하는 것에서부터 크게는 인명 피해까지도 발생시킬 수 있는 매우 심각한 결함 중 하나이다. 이러한 병행성 결함은 비결정적인 특징으로 인하여 재현이 어려우므로 결함의 원인을 찾는 것도 매우 어렵다. 이를 해결하기 위해 다양한 병행성 결함 탐지 연구가 수행되고 있다.

병행성 결함 탐지와 같이 소프트웨어에 존재하는 결함을 탐지하는 방안을 연구할 때에는 그 방안의 효과를 확인하기 위한 방법으로 실험, 사례 연구 및 조사를 수행하여야 한다. 실험이나 사례 연구와 같은 방법은 경험적인 방법으로 결함 탐지 방안의 효과에 영향을 미치는 특정 요인을 식별하고 연구의 효과에 관한 근거를 마련하는데 도움이 되는 방법이다[1]. 실험적 연구란 경험적 연구 방법의 일종으로, 인과관계와 과정을 조사하는 연구 방법이다[1].

기존의 실험들은 연구 결과를 입증하기 위하여 1) 연구의 목적과 일치하는 실험인지를 확인하고, 2) 연구 목적에 따라 실험 대상이 적절하게 선정되었는지 확인한 후, 3) 실험 대상의 테스트 셋이 연구 결과를 입증하기에 충분한 지 확인하고, 마지막으로 4) 실험 결과의 타당성에 대해 분석한다[2]. 결함 탐지 방안의 경우에는 그 방안이 얼마나 결함을 정확하게 탐지할 수 있는지를 보이기 위하여 결함이 존재하는 소프트웨어를 테스트 대상으로 선정하고 해당 대상에 테스트 셋을 적용한 후 그 결과를 분석하게 된다.

그러나 병행성 결함 탐지 방안의 경우에는 연구의 효과를 분석하기 위하여 주로 이미 알려진 결함에 대한 사례 분석이나 벤치마크를 이용하여 왔다. 실험적 연구의 실험 대상으로 병행성을 가진 소프트웨어를 선정한다 하더라도 결함을 가진 테스트 셋을 개발하는 것이 어려웠기 때문이다. 비결정적인 특징으로 인해 동일한 입력에서도 다른 결과를 보일 수 있기 때문이다.

이에 본 논문에서는 병행성 결함 탐지 방안의 효과를 검증하고자 할 때 적용할 수 있는 실험적 연구 방법을 제안하고, 그 방법을 병행성 결함 탐지 도구에 적용하여 그 효과를 분석한다.

2. 관련 연구

병행성 결함 탐지 방안의 실험적 연구의 대상은 병행성을 가져야 하므로 다중 스레드 혹은 다중 프로세스로 동작하는 소프트웨어이다. 이 때 테스트 셋은 실험 대상 소프트웨어에 존재하는 결함이 된다. 테스트 셋을 확보하기 위한 방법은 크게 3가지로 구분할 수 있다.

첫 번째는 결함이 존재하는 테스트 코드를 직접 작성하는 것이다. 도구가 탐지 가능한 결함을 특정한 후 해당 결함이 존재하는 테스트 코드를 직접 작성하고 도구를 적용할 수 있다. 하지만 이 방법은 테스트 코드

* 교신저자(Corresponding author)
○ 발표자

생성에 대한 시간적 부담이 크고 테스트 코드에 대한 검증 또한 추가적으로 이뤄져야하기 때문에 기존의 실험적 연구에서는 잘 사용되지 않았다.

두 번째는 결함이 존재하지 않는 소프트웨어에 인위적으로 결함을 주입하여 실험하는 것이다. 이 방법은 많은 논문에서 채택되어 수행되며, 효과적인 뮤테이션 연산자를 위한 연구가 꾸준히 지속되어 오고 있다[3]. 병행성 결함 탐지를 위한 뮤테이션 연산자에 대한 연구도 이루어지고 있다[1]. 뮤테이션 연산자를 이용하여 결함을 주입할 경우 일관된 결함의 생성이 가능하며, 결함 생성에 소요되는 시간이 첫 번째 방법보다 상대적으로 적다. 또한 실험하고자 하는 도구에 적합한 결함을 직접 주입함으로써 도구 효과 분석에 용이하다. 하지만 도구가 탐지할 수 있는 결함을 커버할 수 있는 뮤테이션 연산자를 설계하는 것이 쉽지 않으며 설계한 뮤테이션 연산자로부터 수행된 실험이 실제로 도구가 탐지해야 하는 결함인지 비교하는 작업이 추가적으로 필요하다.

세 번째는 이미 결함이 존재하는 소프트웨어를 대상으로 실험하는 것이다. 결함이 존재하는 벤치마크[4]나 실제 결함 사례를 대상으로 하는 이 방법은 수많은 논문에서 채택되어 수행되고 있다[5]. 하지만 실제로는 벤치마크나 어플리케이션의 일부만 사용할 뿐이고, 벤치마크에 존재하는 결함은 특정 도구를 대상으로 작성된 것이 아니기 때문에 도구의 효과를 정확하게 분석하는 것은 쉽지 않을 수 있다.

병행성 결함 탐지를 위한 실험적 연구 논문에서는 벤치마크와 실제 결함 사례를 대상으로 하는 방법을 가장 많이 차용하고 있으나, 본 논문에서는 결함 탐지 방안의 객관적인 비교를 위하여 결함이 존재하지 않는 소프트웨어에 인위적으로 결함을 주입하는 방법을 중심으로 연구한다.

3. 병행성 결함 탐지를 위한 실험적 연구 방법

본 논문에서는 기존의 실험적 연구 방법을 토대로 병행성 결함 탐지 방안의 효과를 분석하기 위한 실험적 연구 방법을 제안한다.

3.1 실험 목적 정의

명확하게 목적을 정의하는 것은 실험 설계에 필수적이다. 본 논문에서 실험의 목적은 병행성 결함 탐지 방안의 효과를 확인하는 것이다. 병행성 결함 탐지 방안의 효과는 해당 방안으로 탐지 가능 결함을, 실제로 탐지해 내는지 여부를 말한다.

3.2 실험 목적에 맞는 대상 설정

실험 대상은 실험이 실행되는 단위이다. 병행성 결함 탐지 방안의 실험 대상은 병행성을 가진 소프트웨어로 선정하여야 한다. 즉 병행성 결함 탐지 방안으로 탐지할 수 있는 결함이 존재할 수 있도록 하는 멀티스레드 혹은 멀티 프로세스 기반의 소프트웨어가 대상으로 설정되어야 한다.

3.3 테스트 셋 선정

병행성 결함 탐지 방안의 효과를 분석하기 위해서는 실험 대상에 병행성 결함이 존재하여야 한다. 테스트 셋은 이러한 병행성 결함이 존재하는 소프트웨어의 집합이다. 일반적으로 실험의 대상 크기가 커질수록, 탐지 가능한 결함의 종류가 많아질수록 실험에 사용될 수 있는 전체 테스트 셋이 커지게 된다. 실험에 사용되는 테스트 셋이 커지면 실험의 양이 증가하므로 비용 역시 증가하게 된다. 따라서 전체 테스트 셋에서 실험 목적에 부합하는 테스트 셋을 선정하는 작업이 필요하다.

본 논문에서는 인위적인 결함 주입을 통한 병행성 결함 탐지 방안의 효과 분석 실험을 제안한다. 이를 위해 실험 대상 소프트웨어에 인위적으로 결함을 주입하고 결함 탐지 방안이 해당 결함을 탐지하는지 확인하기 위한 일련의 과정을 통해 테스트 셋을 생성하여야 한다. 이 과정은 그림 1과 같이 결함 식별, 결함 주입, 테스트 셋 선정 단계로 나뉜다.

그림 1결함 주입 테스트 셋 선정 프로세스

결함 식별 단계에서는 표 1과 같이 먼저 병행성 결함 탐지 방안에서 탐지 가능한 결함을 특정하고, 그 결함을 세분화하여 결함 유형을 정의하는 과정이 필요하다. 결함 유형을 세분화하여 정의한 이후에는 각 결함과 관련된 함수를 분류함으로써 결함 주입에 사용한다. 결함 유형이 세분화된 이후에는 각 결함 유형에 따라 뮤테이션 연산자를 정의한다.

결함 주입 단계에서는 주입할 결함과 관련된 데이터를 추출하고 결함 주입 바이너리를 생성하여야 한다. 결함 주입 대상 소프트웨어에는 이미 병행성과 관련된 함수들이 존재하므로 해당 함수의 위치를 추출 한다. 추출된 결함 주입 위치와 뮤테이션 연산자를 맵핑 시켜 결함 주입 코드를 생성한 후, 해당 코드를 빌드 하여 결함이 주입된 바이너리 파일을 생성하여야 한다.

표 1 병행성 결함 유형

구분		결함
데드락	동기화 객체 데드락	동기화 객체 데드락
	통신(메시지) 데드락	동기화 메시지의 수신 실패
		동기화 메시지의 처리 실패
		동기화 메시지의 전송 실패
		동기화 메시지 전송 중 손실
		비동기 메시지 전송 실패
		비동기 메시지 처리 실패
원자성 위반	변수사용 원자성 위반	변수 사용 원자성 위반
	메모리 Read/Write 원자성 위반	공유 메모리에 대한 읽기 연산 수행 중, 다른 프로세스에서 쓰기 연산 수행
		공유 메모리에 대한 쓰기 연산 수행 후 읽기 연산 수행 도중, 다른 프로세스에서 쓰기 연산 수행
		공유 메모리에 대한 읽기 연산 수행 후 쓰기 연산 수행 도중, 다른 프로세스에서 쓰기 연산 수행
		공유 메모리에 대한 쓰기 연산 수행 중, 다른 프로세스에서 읽기 연산 수행
		공유 메모리에 대한 쓰기 연산 수행 중, 다른 프로세스에서 쓰기 연산 수행
순서 위반	생성되지 않은 자원 사용	생성되지 않은 객체 사용
		생성되지 않은 공유 메모리 사용
	초기화되지 않은 자원 사용	초기화되지 않은 객체 사용
		초기화되지 않은 공유 메모리 사용
	해제된 자원 사용	해제된 객체 사용
		해제된 공유 메모리 사용
	공유메모리 접근 순서 위반	공유 메모리에 대해 서로 다른 프로세스에서 각각 읽기, 쓰기 연산 수행
		공유 메모리에 대해 서로 다른 프로세스에서 모두 쓰기 연산 수행

테스트 셋 선정 단계에서는 병행성 결함이 주입되어 결함 탐지 방안이 적용되기 전 원본 소프트웨어의 동작 상태를 모니터링함으로써 테스트 케이스를 생성한다. 주입한 결함 코드가 실행될 수 있는 입력을 찾아, 소프트웨어의 동작 상태를 모니터링하고 결함 주입 코드가 실행되는 경우를 테스트케이스로 생성한다. 주입한 결함 코드의 실행 여부는 결함 주입 바이너리 생성 단계에서 추가한 테스트 출력문이 소프트웨어 실행 시 출력되는지 확인함으로써 알 수 있다. 이렇게 생성된 테스트 케이스의 집합들이 테스트 셋으로 선정된다.

3.4 실험 수행

대부분의 결함 탐지 방안은 결함을 탐지하였을 경우 정보를 수집하고, 수집된 내용을 분석하여 결함 탐지 여부를 확인한다. 실험 수행 단계에서는 3.3에서 생성된 테스트 셋에 병행성 결함 탐지 방안을 적용하여 결함을 탐지하는지를 확인한다. 이 때 먼저 확인되어야 할 것은 결함 탐지 방안으로 인한 영향이다. 병행성 결함은 비결정적인 특성을 가지며, 출력문 하나에도 실행 흐름이 변경될 수 있기 때문이다. 따라서 결함 탐지 방안으로 인한 영향을 확인하기 위해서 먼저 결함 탐지 방안의 적용 전과 후의 소프트웨어 동작 상태를 모니터링한 후, 결함 탐지 방안 적용 전 동작 상태와 동일한지 확인한다. 동작 상태가 일치하지 않을 경우, 결함 탐지 방안으로 인한 변화일 가능성이 있으므로 실험을 중단하고 결함 탐지 방안을 재검토 하여야 한다.

3.5 실험 타당성 분석

실험의 타당성은 실험 수행 결과의 신뢰성에 대한 척도이다. 실험을 계획하는 단계에서 타당성에 대한 위협을 파악하고 그 영향을 평가해야 한다. 실험의 타당성 분석 시 실험의 대상이 실험 목적에 맞게 설정되었는지, 테스트 셋의 크기가 충분한지 확인해야 한다. 다시 말해 실험을 통해 실험의 목적이 입증될 수 있는지 알아보는 것이다.

본 논문에서는 결함이 존재하지 않는 소프트웨어에 인위적으로 결함 주입하는 실험 방안을 제안하였다. 이 때의 테스트 셋은 결함이 주입된 코드이므로 결함 탐지 방안의 적용 시에도 실행 여부가 확실하다. 따라서 테스트 셋으로 선정된 테스트 케이스만으로도 도구의 효과를 검증할 수 있다.

4. 사례분석

본 장에서는 POSIX 기반의 병행성 결함을 탐지할 수 있는 도구인 Helgrind[6]와 멀티스레드 C/C++ 프로그램에서의 병행성 결함을 탐지하기 위한 도구인 DRD[7]를 실제 적용시켜 도구의 효과를 분석한다. 두 도구의 동작 환경 및 탐지 가능한 결함은 표 2과 같다.

표 2 병행성 결함 탐지 도구의 비교

항목	Helgrind	DRD
탐지 가능 결함 유형	-POSIX pthreads API의 잘못된 사용 -데드락 -데이터 경합	-POSIX pthreads API의 잘못된 사용 -데이터 경합 -잠금 경합

두 도구에서 병행성 결함을 인위적으로 주입할 소프트웨어는 표 3와 같다. 두 소프트웨어 모두 병행성 결함이 발생할 수 있는 동기화 관련 함수가 존재하며, 멀티스레드로 실행이 가능하여 실험 대상으로 선정하였다.

표 3 실험 대상 소프트웨어

소프트웨어	설명	버전	크기
SQLite	데이터베이스	3.24.0	8.5MB
BRL-CAD	솔리드 모델링 시스템	7.26.4	188MB

실험에 사용할 테스트 셋은 표 1의 병행성 결함 유형에 Markus Kusano와 Chao Wang이 제안한 병행성 뮤테이션 연산자를 적용하였다[8]. 그 결과 SQLite의 경우 교착상태와 관련된 테스트 셋 18건, 순서위반과 관련된 테스트 셋 11건으로, 전체 29건의 테스트 셋이 선정되었다. BRL-CAD의 경우 교착상태와 관련된 테스트 셋 12건, 원자성 위반과 관련된 테스트 셋 60건, 순서 위반과 관련된 테스트 셋 14건으로, 전체 86건의 테스트 셋이 선정되었다.

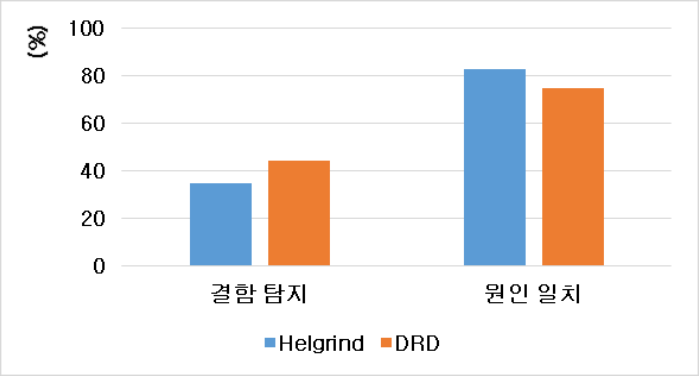


그림 2 병행성 결함 탐지 실험 결과

병행성 결함은 재현 여부가 비결정적이므로 정확한 결함 탐지 여부를 확인하기 위하여 각 테스트 셋 별로 동일한 실험을 5회 반복하였다. 실험 결과는 그림 2와 같이 Helgrind와 DRD가 각각 34.78%, 44.35%의 결함을 탐지하였고, 탐지된 결함 중 각각 82.5%, 74.51%의 결함에 대해 정확한 원인을 진단하였다. Helgrind의 경우에는 SQLite의 29건 결함 중 15건을 탐지하고 그 중 10개의 결함에 대해 정확한 원인을 진단하였다. BRL-CAD에 대해서는 86건의 결함 중 25건의 결함을 탐지하였고, 그 중 23건의 결함에 대해 정확한 원인을 진단하였다. DRD는 SQLite에서 22개의 결함을 탐지하고 그 중 15개의 결함에 대해 정확한 원인을 진단하였다. BRL-CAD에서는 29개의 결함을 탐지하였고, 그 중 23개의 결함에 대해 정확한 원인을 진단하였다.

본 실험을 통하여 Helgrind와 DRD 도구에 대한 병행성 결함 탐지율과 얼마나 정확하게 원인을 진단하는지는 확인하고 비교할 수 있었다.

5. 결론 및 향후 연구

본 논문에서는 병행성 결함 탐지를 위한 실험적 연구 방법을 제안하였다. 기존의 실험적 연구 방법을 기반으로 병행성 결함 탐지를 위한 실험적 연구 방법을 제안하였다. 병행성 결함을 탐지하고 디버깅 정보를 제공하기 위한 결함 탐지 방안의 효과를 입증하기 위해 본 논문에서 제안한 실험적 연구 방법을 이용할 수 있다.

향후에는 제안하는 방안을 실제 병행성 결함 탐지 방안에 적용하여 그 효과를 분석할 예정이다. 또한 제안하는 방안은 병행성 결함뿐만 아니라 소프트웨어의 다양한 결함(메모리, 예외처리, 성능 등)을 탐지하는 것을 목적으로 하는 실험에도 적용할 수 있다. 따라서 병행성 결함이 아닌 다양한 소프트웨어 결함 탐지 방안에도 본 논문의 실험적 연구 방법을 적용하여 그 효과를 확인하는 연구를 진행할 예정이다.

6. 참고문헌

[1] Melo, Silvana M., et al. "Empirical Research on Concurrent Software Testing: A Systematic Mapping Study." Information and Software Technology (2018).
[2] Kasikci, Baris, Cristian Zamfir, and George Candea. "RaceMob: crowdsourced data race detection." Proceedings of the twenty-fourth ACM symposium on operating systems principles. ACM, 2013.
[3] Natella, Roberto, Domenico Cotroneo, and Henrique S. Madeira. "Assessing dependability with software fault injection: A survey." ACM Computing Surveys (CSUR) 48.3 (2016): 44.
[4] Thomson, Paul, Alastair F. Donaldson, and Adam Betts. "Concurrency testing using schedule bounding: an empirical study." ACM SIGPLAN Notices 49.8 (2014): 15-28.
[5] Khoshnood, Sepideh, Markus Kusano, and Chao Wang. "ConcBugAssist: constraint solving for diagnosis and repair of concurrency bugs." Proceedings of the 2015 international symposium on software testing and analysis. ACM, 2015.
[6] Helgrind, http://valgrind.org/docs/manual/hg-manual.html.
[7] SQLite, https://www.sqlite.org/index.html
[8] Markus Kusano, and Chao Wang. CCmutator: A Mutation Generator for Concurrency Constructs in Multithreaded C/C++ Applications. Proceedings of the 28th IEEE/ACM International Conference on Automates Software Engineering, pp.722-725, 2013.

항공기 프로펠러 고장진단을 위한 압력센서 비교

송슬기 O[1], 김은진 [1], 이선아 [1,2]

경상대학교 항공우주 및 소프트웨어공학전공 [1]

경상대학교 정보과학과 [2]

ssyyuu257@naver.com, asd0755@naver.com, saleese@gnu.ac.kr

Comparison of Pressure Sensors for Failure of Aircraft Propeller

Seul-Gi Song O[1] · Eun-Jin Kim[1], Seonah Lee[2]

Department of Aerospace and Software Engineering[1]

Department of Informatics[2]. Gyeongsang National University

요 약

본 연구에서는 항공기(Aircraft)의 안정적인 비행을 돕기 위해 프로펠러에 압력센서를 부착하여 프로펠러의 각도가 4.5°를 중심으로 바람의 저항에 맞게 각도를 유지하지 못하거나 벽과의 충격 또는 고장으로 인한 프로펠러의 변형이 감지되면 모니터를 통해 고장유무를 판별하기 위한 압력센서를 비교 실험한다. 이를 통해 항공기(Aircraft)의 프로펠러의 고장 유/무를 판별, 진단하여 비행시 예방책을 모색하는데 도움을 주고 메인 몸체의 고장을 방지하며 인명피해 및 자원적 손해를 예방할 수 있도록 돕는다.

1. 서 론

항공, 기계, 발전, 화학공정 등 결함이나 손상이 발생할 경우, 시스템 가동 중단을 야기할 수 있으며 이로 인한 사회적, 경제적 피해의 규모가 막대한 분야에서 고장진단 시스템은 중요하다.

고장진단 시스템을 도입하여 고장이 발생하기 전 시스템을 구성하는 부품의 정상 유/무와 고장 발생 징후를 진단하여 고장 발생률을 낮추고 불필요한 부품 교체로 인한 비용 손실을 최소화시키기 위해 고장진단 및 고장예측 기술이 활발히 연구되고 있다[1, 2, 3].

고장진단 기술을 개발하기 위해서 시스템 고장상태를 만든 후, 고장상태인 대상의 데이터 값을 수집하여 고장상태에 따른 예방책을 모색할 수 있다. 또한 센서를 통해 시스템의 상태를 실시간으로 감지하고 시스템의 이상을 조기에 예측하여 미래에 발생할 고장을 예측 및 방어한다.

무인기의 경우 평균적인 유인항공기와 다르게 프로펠러의 로터 블레이드(rotor blade)[가 허브(hub)와 일체형으로 4.5 각으로 고정되어 있다. 이때 추락과 벽, 물체로 인한 충격이 발생하게 되면 무인기의 경우 프로펠러가 일체형이므로 로터 블레이드의 변형으로 비행이 불가능하거나 정확한 조종이 불가능하게 된다[7, 8].

본 논문에서는 항공기의 로터 블레이드 즉, 프로펠러에 압력센서를 부착하여 프로펠러의 휘어짐을 인식하여 조종자에게 상태 알림을 통해 2차 사고를 방지하고 프로펠러의 부러짐과 꺾임을 예방하기 위해 날개에 압력센서를 사용하여 프로펠러 충돌과 꺾임, 휘어짐, 찢어짐이 감지되면 조종자에게 알림을 알리고 고장발생 유/무를 판별하여 안전한 비행 상태인지 판단할 수 있도록 돕는다.

2. 배경 사건

실제 항공기 프로펠러 고장으로 발생한 사고들이 있다. 먼저 수리온 헬기 고장 사고가 있었다 [4]. 해당 사고에서 프로펠러는 회전날개 한 곳의 접이장치에 문제가 생기면서 회전 날개가 먼저 떨어져 나갔으며, 이로 인해 나머지 3엽이 통째로 떨어져 나가는 참사로 이어졌다. 사고의 원인으로는 접혔던 블레이드를 완전히 펼쳤는지 확인을 요구하고 있다. 또한 엔진진동을 완충하지 못하여 로터 블레이드가 휘어지거나 스트레스 축적되어 부서지거나 꺾임이 발생할 수 있음을 보고 있다.

다음 마리온 헬기의 추락 사고가 있었다 [5]. 이 사고의 원인을 분석해 보면 이륙시 동력 엔진의 진동이 커지면서 회전날개에 충격이 전달되었다는 경우로서 기어박스나 진동저감장치의 문제가 아닐 것으로 예측하고 있다. 먼저 회전날개 1엽이 떨어져 나간 뒤 나머지 회전날개 3엽이 통째로 떨어져 나간 사실을 비추어 수리온에 비해 접이장치가 더 튼튼하므로 접이장치의 고장은 아닐 것이라고 보고 있으나 접히는 부분에 추가적인 구조 설계가 변경되고 부품이 많아지면서 고장 발생소지가 더 있을 수 있을 것으로 보고 있다

3. 실험 계획

표1. 실험 횟수

센서	정상 상태 휘어짐 실험 (가로 or 세로)	가로 찢김	세로 찢김	절단
압력센서 FSR 0.58" 원형	가로 10 번 실험 세로 10 번 실험	좌 우 각 2번씩 찢음 4X10 = 40회	2번씩 찢음 2X10 = 20회	총 2번 2X10 = 20회
압력센서 FSR 1.5" x 1.5" 사각형	가로 10 번 실험 세로 10 번 실험	좌 우 각 4번씩 찢음 8X10 = 80회	좌 우 각 4번씩찢음 8X10 = 80회	총 2번 2X10 = 20회
압력위치센서 FSLP 4.0" x 4.0" 직선형 (17cm)	1cm기점으로 각 5번 실험 16X5 = 90 회	2cm 간격으로 5번 찢음 8X5 = 40회	5cm간격으로 3번 찢음 3X5 = 15회	5cm간격으로 3번 절단 3X5 = 15회
SoftPot Membrane Potentiometer	토트 기점으로 (토트 수 = 21개) 각 5번 실험 21X5 = 105회	2cm 간격으로 4번 찢음 4X5 = 20회	2cm간격으로 4번 찢음 4X5 = 20회	2cm간격으로 3번 절단 3X5 = 15회

항공기의 프로펠러의 정확한 구동을 위해 사전 점검과 사후 점검이 이루어진다. 단 그 과정은 사람의 육안으로 확인하므로 정확도가 높다고 볼 수 없다. 그러므로 작동하는데 문제가 없는지 정확한 수치를 확인하기 위해 압력센서, 충격센서, 휘어짐센서 등 이용하여 고장을 예측할 수 있다.

본 논문에서는 4가지 압력센서를 사용하여 프로펠러를 대상으로 고장진단을 실험하고 고장을 예방하기 위해 임의적으로 센서를 꺾고 이상 각도치의 휘어짐, 충격 그리고 찢어짐을 실험하여 고장 유/무를 판별할 계획이다.

드론의 프로펠러의 규격은 일반적으로 APC형식과 GWS형식이 있다. 어떤 형식이든 직경x피치(단위 : 인치)로 나타내며, 피치가 커질수록 한 바퀴 회전할 때 전진하는 거리가 크기 때문에 비행속도가 빨라진다. 이러한 원리에 따르며 직경이 커질수록 힘이 좋아지며 피치가 커질수록 속도가 빨라지는 원리를 가지고 있다.

이러한 원리를 인지하고 본 실험에서는 APC형식의 드론 프로펠러를 대상으로 고장진단을 위한 압력센서를 비교 실험하였다. APC형식의 각도는 4.5도이고 회전 시 프로펠러의 뒤틀림이나 곡선의 휘어짐은 거의 없지만 비행 실수로 프로펠러가 벽과 자주 충돌하는 상황이 발생한다. 이 상황에서 프로펠러의 꺾임이나 뒤틀림, 깨짐을 육안으로 판별하기 어려워서 고장진단이 필요하다.

본 실험은 APC 드론 프로펠러의 고장 진단이라는 특정 상황을 가정하여 실험한다. 단, 프로펠러 자체에 압력센서를 부착하여 외부의 압력을 가한 것이 아닌 임의적으로 실험자가 압력센서의 상태를 조작하여 다양한 상태에 따른 압력센서의 결과값을 조합하여 고장진단의 결과를 도출한다. 압력센서의 실험 횟수는 표1과 같다. 예를 들어 첫번째 압력센서 FSR 0.58'' 원형센서를 실험하기 위하여 휘어짐에 대한 가로, 세로 각 10번씩 총 20번, 가로 찢김 40회, 세로 찢김 20회, 절단 20회, 총 100회의 실험을 진행하였다.

이러한 실험을 통하여 FSR 원형센서, FSR 사각형센서, FSLP 직선형센서, SoftPot Membrane Potentiometer 센서를 이용하여 최소의 각도로 휘어질 때 인식되는 각도를 측정하고 압력센서에 충격을 주었을 때 인식이 되는지를 확인한다. 찢어진 상황에 따라 각 센서들의 출력값을 측정하여 항공기 프로펠러에 적용할 수 있는 센서를 구별하고 적합성을 비교하여 결과를 보고하고자 한다.

4. 실험 결과

실험 결과를 정리한 표 2을 보면 4가지 센서 모두 충격에 대한 반응도는 높은 편이다. FSR 원형 센서의 경우 20° 휘어짐에도 압력이 측정 가능하며 가로의 찢어짐인 경우에는 정상적인 압력을 출력하여 정상작동하는 모습을 알 수 있었다. 그러나 세로 찢어지거나 절단 된 경우에는 전체적인 압력은 45% 압력 수치값이 감소된다. 이때 찢어짐에도 불구하고 휘었을 때 압력을 감지할 수 있었다.

FSR 사각형 센서의 경우 40°~90° 휘어짐에 압력이 측정 가능하며 가로 찢어짐의 경우 찢어진 기점으로 찢어지기 전 안쪽 부분은 정상 수치가 도출된다. 반면 찢어지 기점 바깥쪽은 12% 압력 수치값이 감소하였으며 휘었을 경우 정상일 때 900이상이 측정되었던 값이 700이하로 측정되었다. 그리고 절단된 경우는 가로 찢김과 같이 바깥쪽만 60%의 압력이 감소되었다.

FSLP 센서의 경우 45°~90°로 휘어졌을 때 압력이 감지 되었으며 가로 찢은 경우 비정상적인 압력과 위치 결과값을 출력하였으며, 세로 찢김은 압력은

표2. 압력 센서의 비교

센서	인식되는 휘어짐 각도(°)	휘어진 위치 추적	충격의 정확도	찢김				파손 시 센서도 같이 고장
				가로 찢김	세로 찢김	휘어짐	절단	
압력센서 FSR 0.58" 원형	20°	X	높음	정상작동	45% 압력 수치값 감소	40% 압력 수치값 감소	전체적으로 감지 가능, 45% 압력 수치값 감소	△
압력센서 FSR 1.5" x 1.5" 사각형	40°~90°	X	높음	찢긴 부분 안쪽은 정상 수치값 출력, 찢긴 바깥쪽은 12% 수치값 감소	찢어진 열에는 45% 압력 수치값 감소됨, 그러나 찢어지지 않은 열은 정상값 출력	0~750까지 측정가능	절단된 부분 기점으로 안쪽은 정상 압력값 출력, 바깥쪽은 60% 감소	△
압력위치센서 FSLP 4.0" x 4.0" 직선형	45°~90°	○	높음	비정상적인 압력값과 위치 출력	정상적인 압력값 출력, 단 위치와 압력의 초기값이 0으로 시작하지 않음	위치와 압력을 출력하는데 문제는 없으나 초기값이 0으로 시작하지 않음	센서 고장 발생	△
SoftPot Membrane Potentiometer	30°	○	높음	센서 고장 발생	센서 고장 발생	압력값, 위치값 출력 안됨	압력값, 위치값 출력 안됨	○

표3. 압력 센서의 장단점 비교표

센서	장점	단점
압력센서 FSR 0.58" 원형	좁은 범위에 각 센서를 부착하여 고장 유/무를 판별하기 적합	고장위치 정보를 알 수 없음
압력센서 FSR 1.5" x 1.5" 사각형	휘어짐과 꺾임, 충격의 출력값이 위치와 상관없이 모두 일정하게 출력	고장위치 정보를 알 수 없음
압력위치센서 FSLP 4.0" x 4.0" 직선형	넓은 범위로 압력과 위치 확인 가능 찢어짐에 대한 출력 가능	불규칙적인 값의 출력
SoftPot Membrane Potentiometer	압력과 위치를 확인할 수 있음	압력의 변화에 따라 위치 값이 변동 고장 시 압력, 위치 출력 불가능

출력가능했으나 위치의 초기값이 0이 아닌 다른 값으로 초기값이 잡혔다. 휘어짐 또한 초기값이 0으로 잡히지 않았으며 절단된 경우에 센서가 고장나는 현상이 발생하였다.

SoftPot Membrane Potentiometer 센서는 30°의 휘어짐에 압력이 인식되었으나 찢어짐 실험을 했을 때 센서 고장이 발생하여 압력값과 위치값 출력이 불가능하였다.

5. 논의

실험 결과를 바탕으로 실험한 압력센서의 장점, 단점을 논의한다.

표 3는 4절에서 프로펠러 고장을 프로펠러 안 펴짐, 구부러짐, 부서짐 총 3가지로 가정하여 휘어짐, 찢어짐, 충격을 임의적으로 주어 실험한 결과를 바탕으로 장단점을 간략하게 표기했다.

원형센서는 크기가 작은 특징이 있으므로 좁은 범위에 여러 개의 센서를 부착하여 고장 유/무를 판별하기 적합한 장점이 있으나 한 개의 센서로는 고장의 위치를 알 수 없다.

사각형 센서는 넓은 면적을 측정하는 장점으로 균일하게 모든 면에서 압력값이 출력되었으나 이 센서 또한 원형센서처럼 한 개의 센서로는 고장의 위치

정보를 알 수 없다.

FSLP 직선형 센서는 압력과 위치의 정보를 출력하며 찢어진 경우에도 낮아진 압력의 값을 출력한다. 그러나 찢어진 시간을 기점으로 초기값이 불규칙하게 선언되므로 안정성에 어긋난다

마지막으로 SoftPot Membrane Potentiometer 센서는 압력의 위치와 정보를 알 수 있다.

마지막으로 모든 센서가 파손 시 센서의 작동이 중지되는 단점이 있다. 우리는 이러한 비교를 통해서 4개 중에서는 SoftPot Membrane Potentiometer 센서가 프로펠러 압력 센서로 사용하여 고장진단하기에 가장 장점이 있음을 확인할 수 있었다.

6. 결 론

본 논문 실험은 외부의 힘에 의한 센서의 변형에 따른 값의 출력을 보고 고장 유/무를 판별하기 위한 압력센서 실험을 수행하였으며 항공기 프로펠러에 고장진단으로 사용하기 적합한 압력센서를 분류하고 선정하였다. 이를 통해 고장진단의 중요성을 인지하고 압력센서의 쓰임을 활용하여 항공기의 프로펠러를 대상으로 고장진단 실험을 해보았다.

이를 통해 시스템 고장에 따른 적절한 조취를 취함으로써 불필요한 유지보수 비용을 줄일 수 있고, 시스템의 안전성과 신뢰성을 높여 인명피해와 고가의 시스템 고장을 예방할 수 있음을 이해했다.

향후 연구로는 현재 센서들의 부족한 점을 보완할 방법을 찾을 계획이다. 예를 들어 압력 센서를 실험한 결과 고장추적은 가능하나 원인은 규명 불가능하였다. 이런 경우, 실제 프로펠러 내에 압력 센서를 삽입 시 꺾임, 휘어짐, 충격인지는 알 수 없으므로 향후 연구에 각 센서의 쓰임을 따로 두어 감지되는 이유와 고장의 유/무를 구별하는 것을 실험하여 고장진단의 정확성을 높이는 논문을 작성할 예정이다.

참고문헌

[1]구윤성, 이성화, 김지환, 이형철, ' Sensor Placement for Verification of the diagnosis algorithm of control surfaces', 2011 한국항공우주학회 학술발표회

[2]임헌영, 허성필, ' Propulsion System Model Based Unmanned Aerial Vehicle State Monitoring and Fault Diagnosis', 2016 한국추진공학회 학술대회논문집

[3]임준완, ' A Study on the Safety of Flight(SOF) Assure through Aircraft Diagnostics Systems', 2017 항공우주시스템공학회지

[4]이철재, "프로펠러 통째 날아간 헬기, 추락사고 전날 날개 부품 교체", 2018 중앙일보신문 http://www.koreadaily.com/news/read.asp?art_id=6391466

[5]노지원, "마린온 추락 원인…기어박스 고장, 회전날개 지지축 ", 2018 한겨레신문 http://www.hani.co.kr/arti/politics/defense/853997.html

[6]" FSR 400 Data Sheet : FSR 400 Series Round Force Sensing Resistor", 압력센서 FSR 0.58" 원형 https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/2010-10-26-DataSheet-FSR400-Layout2.pdf

[7]KAI 에비에이션 센터, '날개의 구조 및 기능/ 프로펠러의 종류와 기능' http://www.kaicamp.com/html/mn03/03010303.php

[8]이종화, '프로펠러의 원리', 2005 The Science Time

대인관계 개선 VR 콘텐츠 생성을 위한 저작도구 설계와 구현

심근웅 ○, 윤인호, 임시연, 이양민, 이재기
동아대학교 컴퓨터공학과
geunwoong0624@naver.com, dbscjfgh10@naver.com, gum4019@gmail.com,
yangwenry@dau.ac.kr, jklee@dau.ac.kr

Design and Implementation of VR Content Authoring Tool for Fence-Mending System

Geun-Ung SimO, In-Ho Yun, Si-Yeon Yim, Yang-Min Lee, Jae-Kee Lee
Dept. of Computer Engineering Department, Dong-A University

요 약

각박한 현대 사회에서 사람들 사이의 갈등이 발생하면 이는 쉽게 회복되지 않는다. 그 이유는 대인관계 개선의 실마리를 찾기도 어렵고 방법도 부족하기 때문이다. 일반적으로 심리 상담은 과거에 갈등이 발생한 시점을 회상하는 방법을 사용하는데 본 논문에서는 이런 회상 방법에 VR 기술을 적용하고자 한다. 이를 위해서 상담자의 사연에 근거한 대인 관계 개선용 VR 콘텐츠를 몰입감 있도록 작성하는 방법이 필요하다. 본 논문에서는 VR 콘텐츠를 저작할 수 있는 저작도구 설계와 구현 방법에 대해 연구하고 실제 저작도구를 개발하였다. 본 논문의 저작도구를 활용하여 대인 관계 개선용 VR 콘텐츠를 쉽게 생성할 수 있고 이는 대인관계의 개선과 사회 전체의 정신 건강 유지에도 도움이 될 것으로 기대된다.

1. 서 론

현재 한국은 경쟁의 심화와 가속화로 매우 각박한 사회가 되었다. 이런 환경에서 가족 간 또는 지인 사이의 심리적 마찰에 따른 작은 갈등부터 시작하여 특정 인물이나 조직에 대한 부정인적 감정이 작용하여 여러 가지 범죄를 일으키는 사회적인 병리 현상도 함께 증가하고 있다. 결과만을 보면 극단적인 갈등이라도 초기에 부정적인 감정을 해소하는 것이 큰 문제를 미연에 방지할 수 있다. 그리고 갈등이 발생한 과거 시점을 회상하게 하는 것은 전문 심리 상담에서도 주로 사용하는 방법이다. 이런 과거의 시점의 회상을 위해 본 논문에서는 현재 대중화되고 있는 VR 기술을 활용한다. 현재 VR 기술은 심리 치료를 위한 분야에도 적용되고 있다[1-3].

본 논문에서는 가정 또는 직장 내에서의 다양한 상황에 따라 발생할 수 있는 갈등과 부정적인 감정의 해소를 위해 상대방의 입장에서 자신을 생각해보거나 타인이 되어볼 수 있는 대인 관계 개선용 VR 콘텐츠를 제작하는 방법과 이를 생성할 수 있는 VR 콘텐츠 저작도구의 구현 방법을 제안한다. VR 콘텐츠는 오해와 갈등이 발생했던 과거 상황을 대상자의 묘사에 맞추어야 하며, 이를 위해 정교하고 유연한 콘텐츠 저작은 필수이다. 본 논문에서 제안하는 VR 콘텐츠 저작도구 구현 방법은 대인관계 개선용 VR 콘텐츠를 대상자에 맞춤형으로 생성할 수 있는 방법들을 제시하고 있다. 이 방법을 통해 구현한 VR 콘텐츠 저작도구는 대인관계 개선을 위한 콘텐츠 생성에 유용하여 대인관계 개선에 도움을 줄 수 있도록 한다.

본 논문의 구성은 1장의 서론으로 시작하여, 2장에서 관련 연구를 간략하게 서술하고, 3장에서 저작도구 설계 및 구현 방법과 결과물에 대해 기술하며, 4장은 결론이다.

2. 관련 연구

2.1 VR 콘텐츠 제작용 저작도구 기술

VR 콘텐츠 제작용 저작도구를 대표하는 것은 유니티(Unity)[4]와 언리얼엔진4(UnReal Engine 4)[5]가 있고 근래에 몰입감 높은 VR 콘텐츠 제작에 자주 사용된다[6].

근래에 개발된 VR 기반 편집 솔루션인 나모 360 저작도구도 존재한다[7]. 이 툴은 VR 콘텐츠 개발의 진입장벽을 낮춰 VR 이미지와 VR 동영상을 쉽게 편집하고 편리하게 배포할 수 있도록 돕는 솔루션이다. 이외에도 다양한 VR 콘텐츠 저작도구들이 개발되고 있거나 베타 테스트를 수행하고 있지만 이들의 공통적인 약점은 스토리에 기반한 콘텐츠를 생성하려 할 때 유연성이 부족하다는 점이다. 즉 저작도구에서 기본적으로 제공되는 동작이 아닌 경우, 특정 동작 1개를 만들기 위해서는 애니메이션 전용 개발 도구를 이용하여 1차적으로 개발하고 이를 다시 저작도구에 로딩하여 사용하는 형태가 되어야 한다는 점이다.

2.2 대인관계 개선을 위한 VR 콘텐츠의 특성

사람 간의 갈등 및 심리 치료를 위한 치료 방법 중 과거에 발생했던 문제점을 관찰하고 그 상황을 재현하기 위한 회상 치료법이 있다. 이런 치료법을 VR 기술과 접목한 사례로는 국내에서는 사회공포증을 치유하기 위한 목적으로 삼성에서 개발한 'Be Fearless'가 있다[3]. 국외의 대표적인 예는 미국의 외상 후 스트레스 장애(PTSD) 환자 치료를 위해 개발된 '버추얼 이라크(Virtual Iraq)'가 있다[8][9]. 이런 VR 콘텐츠들은 특정 응급상황을 지속적으로 재생하는 콘텐츠가 대부분이다. 즉 사람 사이의 특정 갈등보다는 괴로워하는 상황을 반복적으로 경험하면서 서서히 그 체험에 익숙해지도록 하는 환경적인 콘텐츠가 현재 주류를 이루고 있다.

대인관계를 개선하는 콘텐츠는 VR의 환경 객체 보다는 상대방이 되는 사람 객체 중심의 제작이 가능해야 한다. 또 사용자의 과거 회상 기억을 최대한 반영할 수 있도록 VR 콘텐츠를 편집할 수 있어야 한다. 여기에 상대방의 음성을 유사하게 제공하면 더욱 몰입감을 높일 수 있고[10][11], VR 콘텐츠 내에서 다른 객체와 상호작용을 할 수 있도록 하면 현실과의 괴리감도 크게 감소시킬 수 있다[12][13].

본 논문에서 제안하려는 VR 콘텐츠 저작도구는 이런 기능들을 충실하게 설계하여 구현하였다. 특히 부부 관계에 초점을 맞추었는데, 부부 사이의 갈등 상황을 매우 유사한 VR 콘텐츠로 반영할 수 있도록 유연한 애니메이션 생성 및 편집 기능을 포함하였다. 또 사람 객체의 표정이나 동작도 사용자의 기억에 맞추어 조정할 수 있도록 하여 대인관계 개선에 초점을 맞춘 콘텐츠를 제작할 수 있도록 하였다.

3. VR 콘텐츠 생성을 위한 저작도구 개발 방법

3.1 VR 콘텐츠 저작도구의 구성

본 논문에서 제안하는 VR 콘텐츠 저작도구의 핵심은 사용자의 스토리텔링에 맞추어 과거를 회상할 수 있는 대인관계 개선용 VR 콘텐츠를 생성할 수 있는 유연한 기능을 갖는 것이다. 이를 위해 VR을 콘텐츠 저작도구는 장면 랜더링 모듈(Scene Rendering Module)과 스토리텔링 모듈(Storytelling Module)로 구성하였다.

3.2 장면 랜더링 모듈

장면 랜더링 모듈은 시뮬레이터와 유사한 큐잉 프로세스를 처리하는 랜더기(Renderer)이다. 랜더링 도중에 사용자가 필요한 정보를 입력할 수 있는 단순한 UI를 제공하고, 입력된 정보를 즉시 반영하여 변경 또는 추가된 VR 콘텐츠를 랜더링할 수 있다. 장면 랜더링 모듈은 사용자가 갈등을 초래했던 상황을 필요에 따라 몇 번이고 반복 재생할 수 있도록 구현하여 심리 치유에 필요한 부분을 집중적으로 체험할 수 있도록 하였다.

3.3 스토리텔링 모듈

이 모듈은 사용자의 스토리텔링(기억, 진술, 기타 정보)을 이용하여 사람 객체와 환경 객체 및 음성을 정교하게 배치할 수 있도록 하는 모듈이다. 스토리텔링 모듈의 전체적인 형태는 그림 1과 같다. VR 콘텐츠 저작도구의 핵심 모듈인 스토리텔링 모듈은 다시 스케줄러(Scheduler) 모듈과 뷰어(Viewer) 모듈로 구분된다. 이 두 가지 모듈이 저작도구로서의 기능을 위한 가장 중요한 역할을 수행한다.

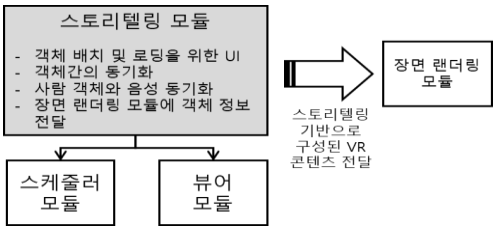


그림 1. 스토리텔링 모듈의 기능 및 동작

3.3.1 스케줄러 모듈

스케줄러 모듈은 일반적으로 사용자가 액션(Action)과 음성을 연결하거나 물건을 잡는(Take) 타이밍 등을 설정하고, 액션의 길이 설정을 위한 모듈이다. 스케줄러 모듈의 구성은 크게 Data, UI, 그리고 스톱워치(Stop Watch)로 구성된다. 그림 2는 스케줄러 모듈의 구성도를 나타내고 있다. 스케줄러 모듈에 특정 내용이 최초로 생성될 때 커서 위치 정보를 초기 위치로 설정하고 Item Dictionary를 생성한다. Item Dictionary는 Action과 Take를 하나의 Item으로 통합하여 관리함으로서 Action과 Take 관리를 분리하지 않고 한 번에 처리할 수 있다. 또한 ListView를 생성하고 Timer의 시간을 0초로 설정하여 초기화하였다. 스토리텔링 모듈의 시작 시점을 0초로 잡고 나머지 객체들의 액션 시간들을 상대적으로 계산할 수 있도록 하였다.

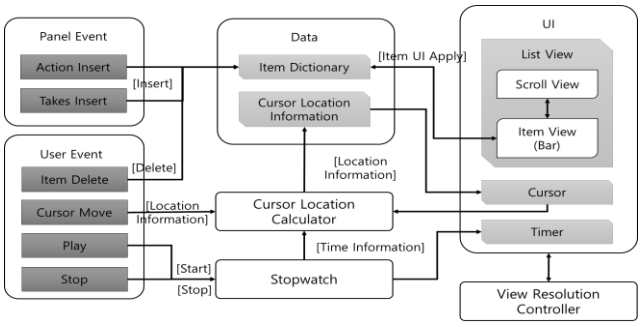


그림 2. 스케줄러 모듈 구성도

외부 패널(Panel)에서 Action과 Take에 대한 정보를 입력하게 되면 Item으로 합쳐 Item Dictionary에 추가하고, Item UI를 생성하여 UI 화면 내에 배치한다. Item UI는 바(Bar) 형태로 되어 있으며 이름, 삭제 버튼, 그리고 시간 정보 바를 포함하고 있다. 그림 3의 하부에 표현된 것이 Item UI이다.

Item View는 그림 2의 우측에 표시되어 있는데, 크게 Item Summary(아이템 요약)와 Item Time Bar(아이템 실행 시간 정보)로 세분화 된다. 아이템 요약은 아이템의 최초 생성 시에 설정 된다. 아이템 실행시간 정보는 애니메이션 바의 크기와 위치에 따라 변화하게 된다. 그림 3에 나타낸 것처럼 흰색의

작은 박스가 하나의 Item에 대한 애니메이션 지속 시간을 의미하는데 크기를 원하는 만큼 조정하여 시간을 설정할 수 있다.

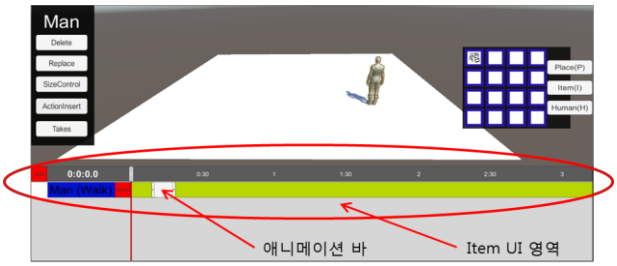


그림 3. 스케줄러에 Item이 추가된 모습

3.3.2 뷰어 모듈

뷰어 모듈은 세 가지 모듈로 구성된다. 인벤토리 생성 모듈, 객체 생성 모듈, 그리고 ItemSetUI 모듈이다.

(1) 인벤토리 생성 모듈

• 인벤토리 생성: 인벤토리는 독립 객체, 액션, 그리고 애니메이션이 포함된 객체 인스턴스를 저장하고 이를 슬롯에 보내주기 위한 저장소이다. 인벤토리 생성을 위해서는 아이템을 먼저 생성하고 연결을 해야 한다.

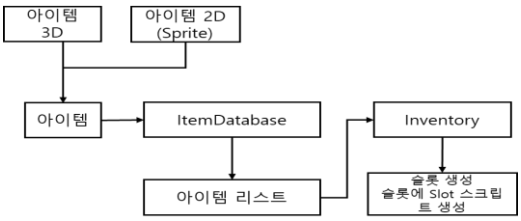


그림 4. 인벤토리 생성의 논리적 과정

• 슬롯 생성: 인벤토리 스크립트 내의 함수가 아이템 리스트들을 로드하여 정보를 가져온다. 인벤토리를 구성하는 슬롯의 x, y 개수, 슬롯의 크기, 그리고 공백의 크기를 지정할 수 있다. 지정한 슬롯의 크기와 공백의 크기에 맞게 자동으로 생성된 슬롯의 위치를 지정하고, 아이템 리스트에 기반하여 슬롯 하나당 아이템이 하나씩 들어가도록 구현되어 있다. 그림 4는 인벤토리 생성을 논리적인 형태로 표현한 것이다.

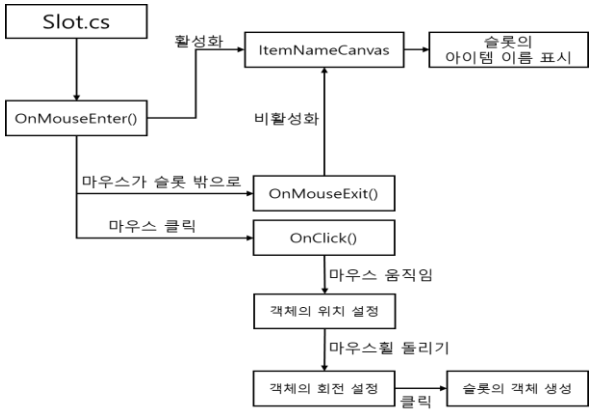


그림 5. 객체 생성 과정

(2) 객체 생성 모듈

객체 생성 모듈은 슬롯 전용 스크립트 파일에 구현되어 있으며 사용자가 이 슬롯에 마우스로 특정 행위를 하면 대응하는 함수가 수행되어 객체를 생성하게 된다. 그림 5는 객체 생성 과정을 도식화한 것이다. 마우스 포커싱, 클릭 등에 따라 각각 다르게 작동하는 UI를 구현하였다.

(3) 아이템 배치 모듈

아이템 배치(ItemSetUI) 모듈은 객체를 배치하는 역할을 담당하며 VR 콘텐츠 저작 도구의 왼쪽에 위치하고 있다(그림 6 왼쪽 참조). 화면상에 생성된 객체위에 마우스를 두거나 객체를 클릭하게 되면 토글 변경 스크립트와 외곽선 설정 스크립트가 서로 연동이 되어 객체의 외곽선 색깔을 변경시키고 이 객체와 사용자가 상호작용할 수 있음을 알려준다. 그림 7은 아이템 배치 모듈의 동작 과정을 요약한 것이다. 아이템 배치 모듈의 스크립트 내에는 VR 콘텐츠를 제작하는 동안 클릭한 아이템의 정보가 저장되며 이 기능들을 통해 VR 콘텐츠를 구성할 각종 객체들을 조작할 수 있다.

그림 6. ItemSetUI 및 ItemName3d 활성화 화면

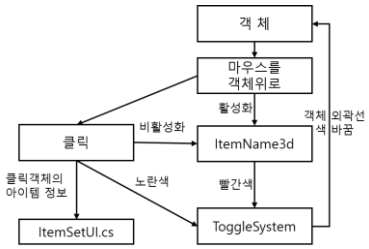


그림 7. ItemSetUI 동작 과정

3.4 VR 콘텐츠용 객체 생성 기법

3.4.1 본리깅 및 페이셜 몰퍼를 통한 표정 제어

본 논문에서 VR 콘텐츠 저작도구 내에서 사용될 사람 객체는 애니메이션 기능을 가질 수 있어야 하는데 모델링만 된 경우는 각각의 팔, 다리, 몸통, 그리고 얼굴 등을 따로 움직일 수 없다. 따라서 사람 객체의 액션을 위해서는 각 부위별로 제어하는 골격과 컨트롤러를 구현하여 연결하는 본리깅(Born Rigging)을 적용하였다. 표정은 대인관계 개선용 VR 콘텐츠를 만들기 위해 필수적인 요소이다. 사람의 표정을 제어하기 위해서 필요한 작업은 얼굴 객체의 정점들을 이동시켜 표정을 만들어내는 '페이셜 몰퍼' 기법과 얼굴 안쪽의 점들을 특정 물체에 맵핑시켜 움직임을 제어하는 '컨트롤러 제어법'을 사용하여 얼굴의 다양한 표정을 만들어낼 수 있다.

3.4.2 애니메이션

사용자의 스토리텔링에 맞춘 특정 동작을 표현하기 위해서 기본 동작을 3D MAX에서 애니메이션으로 미리 작성한다. 또한 VR 콘텐츠 저작도구 내에서 객체의 기본 동작들을 연결하여 다양한 애니메이션을 만들어내는데 그 동작 사이를 부드럽게 연결하는 Idle(정지) 애니메이션도 구현하였다. 유니티에 포함된 메카님을 활용하여 사용자 스토리를 천이도로 구현할 수 있고 이를 유니티 내에서 시간흐름에 따라 재생함으로써 사용자 스토리를 반영하는 콘텐츠를 생성할 수 있다. 또 본 논문의 콘텐츠 특성상 중점을 두고 구현한 것은 사람 객체와 다른 물체와의 상호작용을 컨트롤하는 기능이다. 이 기능은 사용자의 몰입감과 감정 이입을 높여주기 위해 반드시 필요하다. 상호작용이 발생할 객체를 사람 객체가 Take하는 동작을 취할 때 내부 자료 구조를 통해 신체에 연결하는 것으로 상호작용 기능을 구현하였다. 그림 8은 유니티 내에서 애니메이션 작성을 위한 컨트롤러와 상태 천이도의 예시이다. 그리고 그림 9는 VR 콘텐츠 저작도구의 작성 예이다.

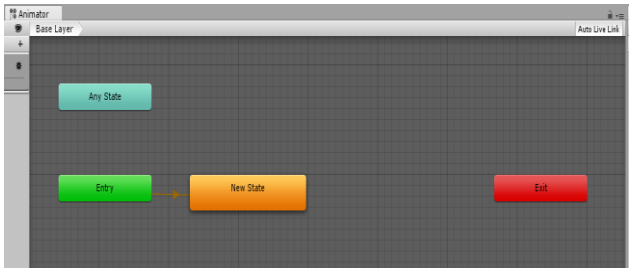


그림 8. 유니티 내의 애니메이션 컨트롤러 상태 천이도

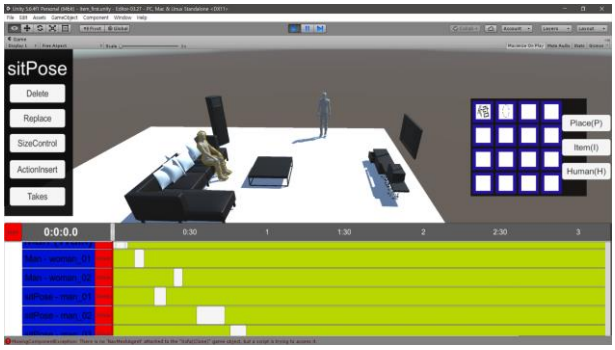


그림 9. VR 콘텐츠 저작도구 및 사용 모습

3.4.3 개발된 VR 저작도구 특징

본 논문의 저작도구는 다른 VR 콘텐츠 저작도구와 비교할 때 사용자의 서술적인 상황 설명을 그대로 반영한 콘텐츠를 제작할 수 있다는 점이다. 단일 사람 객체도 모든 신체 부위를 별도로 제어할 수 있고, 레이어드(layered) 애니메이션을 사용한다. 따라서 사용자가 서술한 완전히 새로운 동작을 표현할 때도 유연하게 대응이 가능하며 기존의 상용 저작도구가 쉽게 만들 수 없는 동작을 구성할 수 있다는 것이 최대의 장점이다. 또한 관계개선에 초점을 맞춘 콘텐츠 제작, 즉 사람의 표정, 음성 표현, 그리고 과거 기억의 충실한 재구성이 가능하도록 저작도구를 구현하였다는 특징이 있다.

4. 결 론

경쟁의 심화로 각박해진 사회에서 동료 사이나 가정 내의 작은 갈등에서부터 시작된 부정적인 감정이 사회를 구성하는 개인은 물론 특정 조직에 범죄를 일으키는 큰 문제로 번지는 경우가 빈번하다. 갈등의 결과만을 보면 매우 복잡하고 극단적인 상황이라 하더라도 갈등 발생 초기에는 의외로 사소한 문제에서 출발하는 경우가 많다. 이런 갈등의 초기에 자신의 갈등 상황을 재현하여 회상하고 3인칭으로 관찰할 수 있도록 하여 갈등의 원인과 이유를 알게 되면 자기 자신은 물론 상대에 대해서도 이해할 수 있게 되어 근본적인 문제 해결이 되는 경우도 있다.

본 논문에서는 이를 위해 상대방의 입장에서 자신을 생각해보거나 타인이 되어 볼 수 있는 대인관계 개선용 VR 콘텐츠 생성용 VR 콘텐츠 저작도구의 설계 및 구현 방법을 제안하였다. 그리고 스케줄러와 뷰어 모듈을 통해 대상자의 스토리텔링에 맞춘 맞춤형 콘텐츠 저작도구 구현 방법을 기술하였다. 제안한 방법을 통해 구현한 VR 콘텐츠 저작도구는 대인관계 개선용 콘텐츠 생성에 적합하며, 실제 관계개선에 효과가 있을 것으로 기대된다. 향후 과제로는 본 저작도구를 심리 치유 현장에 도입하고 실제 효과를 검증하는 것이다.

참고문헌

[1]Health Insurance Review and Assessment service, "Trends in Panic Disorder", 2015.

[2] Stratbase, "VR (Virtual Reality) era, internet traffic rush to prospect", 2016.

[3] U. Y. Yang, “Personal Wearable Display Technology ", Electronic Communication Trend Analysis, Vol 28, No. 5, 2013. 10, pp. 133-144.

[4] Unity : https://unity3d.com/kr

[5] UnReal Engine 4: https://www.unrealengine.com/ko

[6] Statista, “Virtual Reality Head Mounted Displays (HMD) Unit Sales Worldwide from 2014 to 2018,” 2016.

[7] Namo 360 Editor: http://360.namoeditor.com/

[8] Choi Yoon-seop's Healthcare Innovation, 2016

[9] Virtual Iraq: http://www.jba.af.mil/

[10] H.K. Kim, H.S, Kim, J.I. Kim, “Memory space in immersive virtual environment based on VR”. Korea HCI Conference, 2017, pp.1138-1141.

[11] M.H. Shin, E.J. Kim, “A Study on the Methodology of Stereo-Scope Visualization Using Virtual Reality Technique.” Journal of KIICE, 2010, pp. 1482-1487.

[12] J.H. Kim. "A study on an application of 'Virtual Reality Therapy' concerning a technology of real-time interaction." The Korean Society of Cartoon&Animation Studies“, 2011, pp.81-97.

[13] Y.G. Yu, S.J. Kim. “Verification of Social Facilitation Effect in Immersive Virtual Environment”, Conference The HCI Society of Korea, 2018, pp.768-771.

테스트케이스의 제어흐름기반 프로그래밍 실패 피드백 시스템 설계

이성희 ○ 김덕엽 서강복 이우진
경북대학교 컴퓨터학부, 소프트웨어기술연구소
lee3229910@gmail.com, ejrduq77@naver.com, dating1227@gmail.com,
woojin@knu.ac.kr

Design of Programming Failure Feedback System Based on Control Flow of Test Case

Sunghee Lee○ Deok Yeop Kim Seo Kang Bok Woo Jin Lee
School of CSE & SWRC, Kyungpook National University

요 약

프로그래밍 교육 지원을 위한 프로그래밍 채점시스템은 출제자가 문제를 시스템에 공개하고 학습자가 그 문제에 대한 답안 소스코드를 제출하면 그 소스코드를 실행하여 통과, 실패, 에러메시지 등을 알려준다. 그 피드백을 통해 학습자는 소스코드를 디버깅하여 문제풀기를 계속한다. 본교에서도 이러한 채점시스템을 자체 개발하여 진행한 결과 오답판정을 받은 많은 학생들이 자신이 틀린 부분을 정확히 알고 고치지 못하는 학생들이 사실을 확인하였다. 기존의 프로그래밍 채점시스템은 주로 에러메시지를 알려주는 피드백을 진행한다. 따라서 본 논문에서는 프로그래밍 실패에 대한 피드백을 제공하는 시스템을 제안한다. 결함이 있을 수 있는 부분을 알려줌으로써 학습자들이 틀린 부분을 보다 빠르게 찾을 수 있다. 또한 실제 학생이 제출한 답안 소스코드를 예로 들어 본 시스템이 적용 가능함을 보인다.

1. 서론

4차 산업혁명에 관심이 증가함에 따라 프로그래밍 교육에 대한 관심도 함께 증가하고 있다. 이러한 프로그래밍 교육을 지원하기 위한 시스템 중 프로그래밍 채점시스템이 있으며 대표적으로 ACM UVA[1]와 BOJ(BAEKJOON ONLINE JUDGE)[2]가 있다. 일반적으로 이러한 실습 채점 시스템은 다음과 같은 과정으로 서비스를 제공한다. 먼저, 출제자가 문제를 웹사이트 또는 시스템에 공개하고 학습자가 그 문제에 대한 답안 소스코드를 제출하면 채점 시스템은 제출된 소스코드를 실행하여 채점 결과(컴파일 에러, 런타임 에러, 시간초과, 정답, 오답)를 보여준다[3]. 그리고 사용자는 채점된 결과를 보고 본인이 제출한 소스코드가 정답인지 오답인지를 확인하고 오답이라면 소스코드를 디버깅하여 문제풀기를 계속한다.

이러한 채점시스템을 본교 학생들의 프로그래밍 교육을 지원하기 위한 목적으로 자체 개발하였고 현재 프로그래밍 교육에 활용하고 있다[4]. 개발한 채점 시스템을 사용하여 학생들의 답안 제출 이력을 분석한 결과 흥미로운 점을 발견하였다. 많은 학생들이 자신이 제출한 답안이 오답판정을 받았다면 자신이 어떤 부분을 틀렸는지를 정확히 알고 고치지 못하는 경우가 40%나 된다는 것이다. 예를 들어, 표 1은 2018학년도 1학기 기초프로그래밍 수업에 사용된 실습 문제와 해당 문제에 대한 40명의 학생들의 제출이력을 분석한 결과로 약 1/4의 학생들이 반복해서 동일한 문제를 풀지 못하는 것을 알 수 있다.

표 1. 기초프로그래밍 수강생 답안 제출 이력 분석

[문제] 1일째 1원, 2일째 2원, 3일째 4원과 같이, 매일 임금이 이전 날의 2배가 된다고 할 때, 입력 받은 일 수 동안 매일 받는 임금과 받은 임금의 총합을 계산하는 프로그램을 작성하라. (만약 일 수로 0보다 작은 값이 입력되면 “wrong”을 출력하며 다시 입력 받아야 한다.	
1회 이상 틀린 사람	25명/40명 (62.5%)
1회 이상 틀린 사람 중 2회 이상 틀린 사람	10명/25명 (40%)
1회 이상 틀린 사람 중 3회 이상 틀린 사람	6명/25명 (24%)

동일한 문제를 반복해서 풀지 못하는 학생들에게 답안 소스코드가 실행 중에 어느 실행 경로에서 오답이 나는지 알려준다면 학생들이 자신의 답안에서 어느 부분이 잘못되었는지를 보다 빨리 찾을 수 있을 것이다[5]. 따라서 본 논문에서 학생이 제출한 답안 소스코드의 제어흐름도에 프로그래밍 실패 경로를 나타내어 학생들이 자신의 소스코드가 가지는 구조적, 논리적인 결함을 알려주는 피드백 시스템을 제안한다. 또한 실제로 학생이 제출한 소스코드를 사례로 들어 본

시스템이 필요함을 보인다.

2. 관련연구

채점 시스템들은 제출 답안을 자동으로 채점하기 위해 공통적으로 프로그램에 입력될 정보와 미리 정의된 출력 형태를 알려준다. 그러나 제출 답안에 대한 채점 피드백의 지원 방식은 각각 다르다.

대부분의 채점 시스템은 제출 답안을 컴파일하고 실행한 후, 실행결과와 답안을 비교하여 채점한다. 따라서 최소한의 피드백으로 정답(채점 통과), 오답(채점 통과 실패), 컴파일 오류, 런타임 오류 중 해당하는 것을 알려준다. 일부 시스템은 채점 결과를 더 구분하여 어떤 오류인지 또는 어떤 테스트 입력에 대해서 틀렸는지 등 상세한 정보를 알려주기도 한다[6]. 또 학생들이 프로그램의 실행 흐름이나 값의 변화를 이해할 수 있도록 실행 과정을 시각화 한 연구도 있다[7]. 그러나 제출한 코드가 오답인 경우에 어느 부분을 수정해야 하는지에 대해 피드백을 주는 채점 시스템은 없다. 표 2는 채점 시스템들의 채점 통과 유무를 제외한 지원하는 채점 피드백 내용을 나타낸 것이다.

표 2. 채점 시스템 별 채점 피드백 내용

채점 시스템	채점 실패 시, 채점 피드백 내용
ACM UAV	컴파일 오류
BOJ	컴파일 오류
Codeup[6]	컴파일/런타임 오류 메시지, 팁
DOMjudge[8]	컴파일 오류
AOJ[9]	컴파일/런타임 오류 메시지
Programmers[10]	컴파일/런타임 오류 메시지
SPOJ[11]	컴파일/런타임 오류 메시지
KOI[12]	컴파일/런타임 오류 메시지

기존 채점 시스템들의 피드백은 컴파일러나 운영체제가 제공하는 오류 메시지만을 학습자에게 알려준다[8-12]. 이러한 정보들만으로는 학습자들이 작성한 프로그램이 오답일 때 구조적으로 문제가 있는지 또는 논리적으로 문제가 있는지 피드백 할 수 없다. 본 논문에서는 제출된 답안 소스코드의 실행 실패 경로를 분석하여 제출된 답안의 제어흐름이 구조적으로 또는 논리적으로 문제가 있는지를 알려주어 학습자가 프로그램을 보다 신속하게 수정할 수 도와줄 수 있다.

3. 제어흐름기반 프로그래밍 실패 피드백 시스템

3.1 구조 분석을 위한 실행경로 기반의 제어흐름도

일반적으로 제어흐름도는 정적 분석 프로그램 분석 기법으로 컴파일 시간에 가능하다. 하지만 제어흐름도만 사용하여 실제로 프로그램이 어느 경로를 따라 동작할지는 알 수 없다. 따라서 본 논문에서는 제출된 답안 소스코드의 제어흐름도와 실행경로를 결합하여 프로그램이 실제로 수행되는 실행경로 기반의 제어흐름도를 제안한다. 또한 채점 시스템의 문제별 테스트케이스 셋은 모두 문제에서 요구하는 분기 커버리지를 100%를 달성할 수 있다고 가정한다. 또한 그렇지 않은 테스트 케이스 셋은 채점에 사용할 수 없다고 가정한다.

예를 들어, 표 3은 표 1과 동일한 과목의 if문 실습 문항으로 2차원 공간의 x, y 좌표를 입력 받고 그 좌표가 어느 사분면에 존재하는지를 출력하는 실습 문제이다. 이 문제가 요구하는 분기는 총 4개이므로 구조적으로 그림 1(a), 그림 1(b)는 요구하는 모든 분기가 존재하는 제어흐름도이고 그림 1(c)는 필요 없는 분기가 존재하고 그림 1(d)는 필요한 분기가 부족한 제어흐름도이다.

표 3. 기초프로그래밍 if문 실습 문제

[문제]
2차원 공간의 x, y 좌표를 입력 받아 그 좌표가 속하는 사분면을 출력하는 프로그램을 작성하시오. (단, x, y는 0이 아닌 값을 입력한다.)

TC#	Test input	Expected Value
1	3 5	1사분면
2	-3 5	2사분면
3	-3 -5	3사분면
4	3 -5	4사분면

그림 1은 일반적인 제어흐름도와는 다르게 표현되어 있다. 제안하는 실행경로 기반의 제어흐름도는 노드가 녹색(커버된 노드), 흰색(커버되지 않은 노드), 빨간색(2번 이상 중복으로 커버된 노드)으로 표현되어 있다.

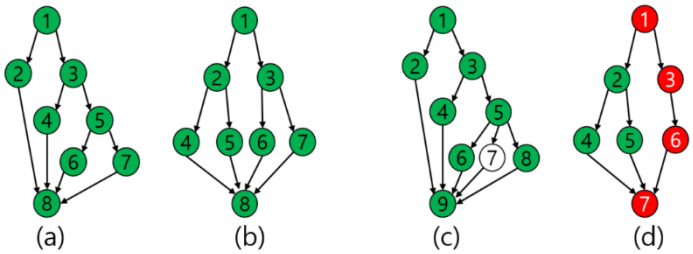


그림 1. if문 실습 문제의 가능한 제어흐름도

예를 들어, 표 3의 테스트 케이스는 문제에서 요구하는 분기 커버리지를 100% 달성할 수 있고 4개의 분기만 커버할 수 있다. 그림 1(c)의 실행경로 기반의 제어흐름도는 커버되지 않은 노드 7을 가지므로 불필요한 분기가 존재하는 것이고 그림 1(d)의 제어흐름도는 노드 1-3-6-7 경로를 2개의 테스트

케이스가 중복해서 커버하기 때문에 필요한 분기가 빠져있다고 말할 수 있다. 이러한 실행경로 기반의 제어흐름도를 사용하여 구조적인 피드백 할 수 있다.

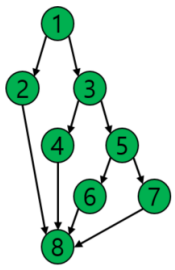
3.2 노드 분석을 위한 실행경로 기반의 제어흐름도

3.1절에서 설명한 표 3의 if문 실습문제를 예를 들어 다시 설명하도록 한다. 그림 1(a)의 구조를 가지는 답안 소스코드의 예를 그림 2와 같이 작성할 수 있다.

```
(전략)
if(x > 0 && y > 0) {
    printf("1사분면");              // 1번 출력문
} else {
    if(x < 0 && y > 0) {
        printf("2사분면")           // 2번 출력문
    } else {
        if(x < 0 && y < 0) {
            printf("4사분면")       // 3번 출력문
        } else {
            printf("3사분면")       // 4번 출력문
        }
    }
}
(후략)
```

그림 2. 기본 블록이 결함을 가지는 답안 소스코드

그림 2의 소스코드는 그림 1(a)의 구조를 가지는 답안 소스코드로 분기 구조는 잘 작성되어 있지만 1-3-5-6-8, 1-3-5-7-8 경로에 해당하는 부분은 채점 결과 오답으로 나온다. 3번 출력문과 4번 출력문을 지나가는 경로의 테스트 케이스들이 오답으로 나오기 때문에 그림 3과 같은 실행경로 기반의 제어흐름도와 프로그래밍 실패 피드백이 함께 나타난다.



TC#1: 1-2-8 Result: Pass
TC#2: 1-3-4-8 Result: Pass
TC#3: 1-3-5-6-8 Result: Failure
TC#4: 1-3-5-7-8 Result: Failure

All nodes are covered. But the failures occur in the paths of TC#3, TC#4.
So you may find a cause or a hint of the failures in the node 1, 3, 5, 6, 7, 8.
Check the nodes carefully.

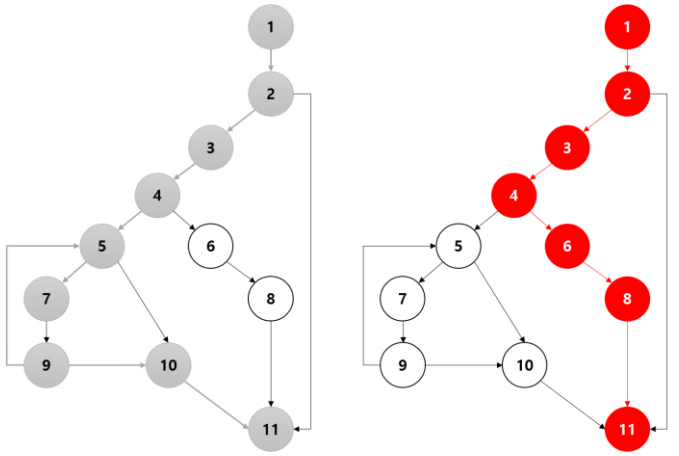
```
(전략)
if(x > 0 && y > 0) {
    printf("1사분면");              // 1번 출력문
} else {
    if(x < 0 && y > 0) {
        printf("2사분면");          // 2번 출력문
    } else {
        if(x < 0 && y < 0) {
            printf("4사분면");      // 3번 출력문
        } else {
            printf("3사분면");      // 4번 출력문
        }
    }
}
(후략)
```

그림 3. 제어 흐름도 기반의 실행 실패 피드백

4. 적용사례

본 장에서 예시로 보이는 소스코드들의 제어흐름도는 표 1에서 나타낸 2018학년도 1학기 기초프로그래밍 수업 내용의 반복문에 해당하는 실습 문제의 한 학생이 제출한 답안 소스코드들이다. 제어흐름도들은 제출된 소스코드의 순서에 맞게 순차적으로 보여주며 제어흐름도에 테스트케이스(TC) 별 커버된 노드는 색으로 강조하였다. 회색으로 강조된 부분은 테스트케이스가 통과한 경로이고 붉은색으로 강조된 부분은 테스트케이스가 실패한 경로이고 강조되지 않은 노드는 커버되지 않은 노드이다.

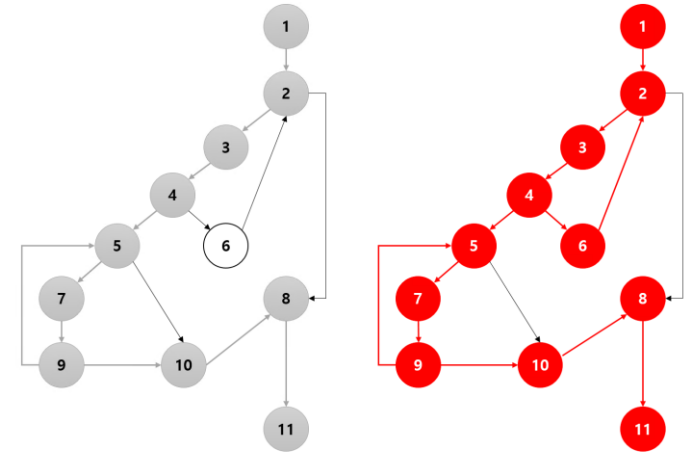
먼저, 학생이 첫번째 제출한 답안 소스코드의 제어흐름도는 그림 4로 나타낸다. 그림 4(a)의 TC#1은 채점 통과했고 그림 4(b)의 TC#2는 채점 실패했다. 실행 결과를 봤을 때, 분기 커버리지는 100% 달성했으므로 분기 구조는 문제가 없음이 확인되었고 실패한 실행 경로를 확인할 필요가 있으며 특히 노드 4, 노드 6, 노드 8을 확인할 필요가 있다고 프로그래밍 실패 피드백을 소스코드 상에 강조할 수 있다.



(a) TC#1을 사용한 소스코드 실행 경로(통과) (b) TC#2 사용한 소스코드 실행 경로(실패)

그림 4. 첫번째 제출된 답안 소스코드의 제어흐름도

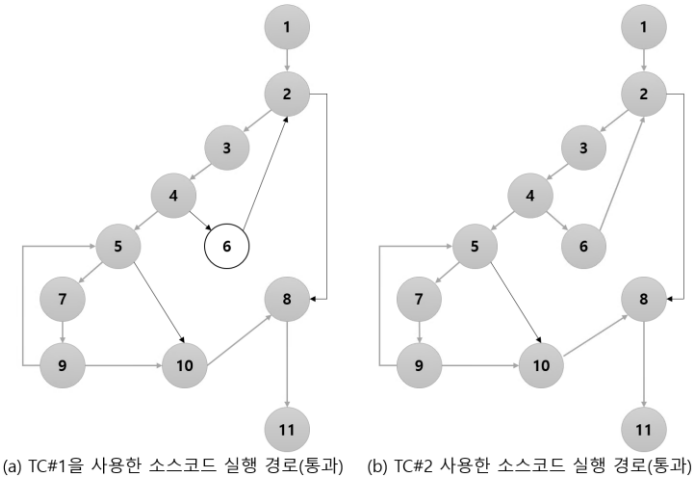
학생이 두번째 제출한 답안 소스코드의 제어흐름도는 그림 5로 나타낸다 그림 5(a)의 TC#1은 채점 통과했고 그림 5(b)의 TC#2는 채점 실패했다. 첫번째 답안 소스코드와 동일하게 분기 커버리지는 100% 달성했으므로 분기 구조는 문제가 없음이 확인되었고 실패한 실행 경로를 확인할 필요가 있으며 특히 노드 4, 노드 6, 노드 10, 노드 8을 확인할 필요가 있다고 프로그래밍 실패 피드백을 소스코드 상에 강조할 수 있다. 두번째 제출된 답안 소스코드를 확인해본 후, 흥미로운 점은 첫번째 프로그래밍 실패 피드백에 해당하는 노드 6과 노드 8에 해당하는 부분이 고쳐졌다는 것이다.



(a) TC#1을 사용한 소스코드 실행 경로(통과) (b) TC#2 사용한 소스코드 실행 경로(실패)

그림 5. 두번째 제출된 답안 소스코드의 제어흐름도

마지막으로 세번째 제출한 답안 소스코드의 제어흐름도는 그림 6으로 나타낸다 그림 5(a)의 TC#1와 그림 5(b)의 TC#2 모두 채점 통과했다. 결론적으로 마지막으로 제출된 답안 소스코드는 정답으로 판정된 소스코드이다. 두번째 제출된 답안과 동일하게 세번째 제출된 답안 소스코드에서도 두번째 프로그래밍 실패 피드백에 해당하는 노드 6에 해당하는 부분이 고쳐진 것을 발견했다. 따라서 본 논문에서 제안하는 프로그래밍 실패 피드백을 참고하면 학습자가 디버깅하는데 도움을 줄 수 있다는 것을 알 수 있다.



(a) TC#1을 사용한 소스코드 실행 경로(통과) (b) TC#2 사용한 소스코드 실행 경로(통과)

그림 6. 세번째 제출된 답안 소스코드의 제어흐름도

5. 결론

본 논문에서는 기존의 프로그래밍 교육을 지원하기 위한 프로그래밍 채점시스템에서 제공하지 않는 프로그래밍 실패에 대한 피드백 시스템을 제안하였다. 실제로 학생이 제출한 답안 소스코드에 적용하여 제안한 시스템이 학습자가 자신이 작성한 소스코드의 결함을 찾아내는데 도움이 될 수 있음을 보였다. 향후 연구로는 제안한 시스템을 실제 사용중인 채점시스템에 적용하고 학습자의 디버깅에 도움을 줄 수 있는 정량적인 근거를 수집하고 피드백 시스템을 확장하고자 한다.

※ 본 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.NRF-2017R1D1A3B04035880)

참고문헌

[1] ACM UVA, https://www.uva.onlinejudge.org
[2] Baekjoon Online Judge, https://www.acmicpc.net
[3] 우여명, 방지웅, 송재민, 유진영, 이상준, “웹 기반 C프로그래밍 언어 학습 시스템의 설계 및 구현”, 한국 정보과학회 컴퓨팅의 실제 논문지, 제20권, 제12호, pp.640-645, 2014
[4] KNU Online Judge, http://selab.knu.ac.kr/training
[5] 박소영, “웹기반 프로그래밍 언어 강의 지원 시스템의 설계 및 구현”, 한국정보통신학회논문지, 제14권, 제12호, pp. 2275-2782, 2010
[6] Codeup, http://www.codeup.kr
[7] Tetsuro Kakeshita, Kosuke Ohta, and Ryo Yanagita, “A Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions”, International Conference on Education Reform and Modern Management, pp.164-167, 2015
[8] DOMjudge, https://www.domjudge.org
[9] Algospot Online Judge, https://www.algospot.com
[10] Programmers, https://www.programmers.co.kr
[11] Sphere Online Judge, http://www.spoj.com
[12] KOI, http://www.koistudy.net

버그 검출 규칙 개선을 위한 코드 문맥 수집 방법에 대한 연구

최윤호 ○, 김석진, 김유진, 권현우, 남재창

한동대학교 전산전자공학부

{uenho830, rkathwks, alwayspring103}@gmail.com, 19yes95@naver.com, jcnam@handong.edu

Mining Context for Detection Rule Revision of a Static Analysis Tool

Yoon-Ho Choi○, Seok-Jin Kim, Yujin Kim, Hyun-Woo Kwon, and Jaechang Nam

School of Computer Science and Electrical Engineering, Handong Global University

요 약

국내 소프트웨어 시장의 규모가 점점 커짐에 따라 소프트웨어의 결함에 대한 효율적인 유지 보수가 중요해지고 있다. 이를 위해 다양한 정적 분석 도구가 출시되어 개발자들을 돕고 있으나, 높은 오탐지 비율이 개발자들 혼란스럽게 하고 유지보수 비용을 오히려 가중시킨다는 문제점이 있다. 이 문제의 해결을 위한 한 가지 방법으로 분석 도구의 오탐지 비율을 최소화하기 위한 결함 검출 규칙 개선이 있다. 본 연구에서는 대표적인 오픈소스 정적 분석 도구인 PMD의 오탐지 결과 사례를 분석하고 규칙을 개선하는데 활용할 수 있는 공통 문맥 정보의 자동 수집 및 수정 가능성을 고찰했다.

1. 서 론

한국 IDC에 따르면, 2017년 국내 상용 소프트웨어 시장 규모는 4조 2850억원에 달하며 2022년에는 5조 2105억원까지 성장할 것으로 예측되었다[1]. 소프트웨어 산업은 꾸준히 성장해왔고 앞으로도 성장할 것이며, 많은 소프트웨어가 개발되고 소비될 전망이다. 이와 같은 상황에서, 소프트웨어를 구매 및 이용하는 소비자들은 더욱 안전하고, 결함이 없는 소프트웨어를 원하는 것은 당연한 일이다. 동시에, 기업 입장에서도 발생하는 결함에 대해 유지 보수가 원활한 소프트웨어를 출시하는 것이 이익을 창출하는데 도움이 된다. 실제로 소프트웨어에 드는 비용 중 소프트웨어 유지 보수에 소요되는 비용이 막대한 비율을 차지한다[2].

이렇게 소프트웨어의 개발 및 유지 보수에 대한 비용과 소요를 절감시키고 효율성의 증대를 위해, 다양한 정적 분석 도구가 출시되어 개발자들을 돕고 있다. 정적 분석 도구의 예를 들면 PMD[3], 파수 Sparrow[4], SonarQube[5] 등이 있다. 이들은 실제 코드의 실행 없이 설정된 규칙에 따라 소프트웨어의 코드를 분석하여 버그를 검출해낸다.

하지만 이러한 정적 코드 분석 도구가 결함이 없는 코드를 결함으로 탐지하여 개발자에게 안내하는 경우가 다수 발생한다. 분석 도구 내의 설정된 규칙이 제 역할을 하지 못하여, 결함이 없는 코드를 결함으로 인식하여 오탐지 결과(false positives)를 다수 발생시키며, 이는 개발자원의 낭비를 야기한다.

이런 경우, 개발자들이 직접 분석 도구의 환경설정을 통해 각자 개인이 사용하기 편리한 맞춤 규칙을 설정하거나, 안내된 정보가 False Positives 인지 True Positives 인지 일일이 확인하며 사용해야 한다. 하지만, 이 과정 자체도 번거롭고, 오히려 개발자의 작업량을 늘리기 때문에 개발 과정의 효율성을 감소시킨다[6].

이러한 문제를 해결하기 위한 한 가지 방법으로, 정적 분석 도구가 오탐지를 최소화하도록 규칙을 수정할 수 있다. 규칙 수정이 정확하게 이루어진 코드 분석 도구를 사용한다면, 소프트웨어 결함 검출과 유지, 보수에 소요되는 비용을 효과적으로 절감할 수 있을 것이다.

본 연구에서는 마이닝 소프트웨어 레포지토리를 통해, 기존의 정적 분석 도구의 규칙을 효율적으로 개선할 수 있는 방법에 대해 연구해 보았다. 이를 위해 오픈소스 정적 분석 도구인 PMD의 오탐지 사례들을 분석하여 버그 검출 규칙을 개선하는데 활용할 수 있는 몇 가지 코드의 공통 문맥(Context)이 존재함을 관찰하였다. 이 결과를 바탕으로 GitHub 등의 소프트웨어 레포지토리에서 버그 검출 규칙 개선에 기여할 수 있는 코드 문맥 정보 자동 수집 가능성을 고찰하였다.

2. PMD 오탐지 결과 분석 절차 및 결과

오탐지 결과 분석을 위해 대표적인 오픈소스 정적 분석 도구인 PMD를 사용하였다[3]. PMD는 대표적인 오픈소스 정적 코드 분석 도구로, 일반적인 프로그램의 결함을 찾아내며 JAVA, JAVAScript, Salesforce.com Apex와 Visualforce, PLSQL 등에 적용 가능하다. 결함을 찾아낼 때에는, 일련의 규칙에 근거하여 코드에 결함이 있는지, 없는지 판별하여 개발자에게 알려준다. 이 도구를 이용하여 오탐지 결과를 수집하고 분석한 절차는 다음과 같다.

2.1 정적 분석 도구 적용

PMD는 ErrorProne이라는 범주 안에 총 97개의 규칙을 제공

표 1. PMD 오탐지 분석 결과

규칙	적용한 프로젝트	총검출수	오탐지수	False Positives의 원인 유형
CompareObjects WithEquals	Incubator -dubbo	4	4	(1)Object/List/Exception/Type type간의 비교, (2) Mapping에서의 key 값에 의한 data 비교, (3) 비교되는 두 항이 정의됨에 있어 서로 연관이 있는 경우, (4)method name에 set/add/store/equals라는 단어 포함한 method 내에서 비교, (5)비교하는 line 밑에 바로 return; 수행, (6)비교항의 변수 이름에 old/new/current가 포함된 경우
	Tomcat	65	65	
Dataflow Anomaly Analysis: DD-Anomaly pattern	jackrabbit	72	61	(1)Null/0으로 변수를 선언하는 경우, (2)2개 이상의 if-else 문/switch문 안에서 개별적으로 변수를 중복 선언하는 경우, (3)for/while문 안의 조건문에서 변수가 특정 조건에 따라 계속 업데이트 되어야 하는 경우 (4)하나의 배열에서 서로 다른 index에 값을 할당하는 경우
	open meetings	8	8	

하며[7], 각 규칙을 오픈소스 소프트웨어의 최신 소스 코드와 1년 전 소스 코드에 적용하여 결함 검출 결과값을 산출하는 과정을 거친다. 초기 단계의 본 연구에서는 분석 도구를 적용하는 과거 코드와 현재 코드의 격차를 충분히 주기 위해 1년이라는 임의의 기간을 설정하였다. 과거의 소스 코드와 현재 소스 코드에서의 결함 검출 결과값을 도출한 이유는 오픈소스 소프트웨어의 담당 개발자들이 1년 동안 오류를 일으키거나 오류가 될 만한 결함을 수정했을 것이라는 가정에 기반하여, 두 코드의 결함 검출 결과값을 비교한 후, False Positive를 1차적으로 구분하기 위함이다. 이렇게 검출된 False Positive는 후에 검출 규칙을 수정하기 위해 사용된다. 이 방식은 FeeFin의 Feedback-based detection rule design과 유사하다[8].

이 과정을 거치면 과거 코드와 현재 코드에 대해 PMD 적용 결과값을 비교할 수 있는 상태가 되며, 비교 결과를 세 가지 경우로 나뉜다. 첫번째 경우는 과거 코드에 결함으로 검출된 결과가 현재 코드에서 사라진 경우이다. 이 경우는 1년 동안 결함을 수정한 경우이므로, 1차적으로 True Positive로 해석한다. 두번째 경우는 과거 코드에 결함으로 탐지된 결과가 현재 코드에서도 그대로 검출되는 경우이다. 이 경우는 1년간 오류를 일으키지 않아 유지된 코드로 보고 False Positive 의심 결과로 해석한다. 세번째 경우는 과거 코드에 결함으로 탐지되지 않은 항목이 현재 코드에서 탐지되는 경우이다. 이 경우는 과거 코드에서 새로운 내용이 추가되어 현재 코드에 있는 결함으로 탐지된 경우이므로 논외로 한다.

2.2 오탐지 결과를 야기하는 원인 분석

1차적으로 True Positive와 False Positive 의심 결과를 구분한 다음, 2차적으로 결함 탐지 결과를 다시 한 번 검토하는 과정을 거쳤다. 이는 True Positive가 확실한지 False Positive 의심 결과들은 정말로 오탐지 결과가 맞는지 확인할 뿐만 아니라 오탐지 결과를 야기하는 원인을 분석하기 위함이다.

이와 같은 확인 절차로 확실하게 False Positive로 분류된 결과를 분석하며, 오탐지 결과가 규칙의 취약점으로 비롯된 몇 가지 사례들을 관찰할 수 있었다. 또한 False Positive 원인 분석 중 규칙의 취약점이 소스 코드 상의 공통된 문맥에서 작용한다는 점도 발견하였다. 코드 상의 문맥이란, 결함이 있다고 안내된 코드를 포함하는 여러가지 맥락을 의미한다. 예를 들어, 결함으로 탐지된 라인이 포함된 method의 이름, 선언되어 있는 변수의 타입, 변수의 데이터 흐름, 예외 처리(Exception Handling) 방법 등을 의미한다.

표 1에는 CompareObjectWithEquals 규칙과 Dataflow - AnomalyAnalysis 규칙의 DD-Anomaly pattern을 적용하여 검출된 모든 안내 정보의 수와 False Positive 검출 횟수, 그리고 이를 야기하는 원인을 정리하였다. 표 1의 원인 유형을 살펴보면 오탐지 결과의 원인이 특정한 문맥을 가진 코드들에 존재함을 확인할 수 있다.

2.3 분석 결과: 검출 규칙 개선을 위한 문맥 정보

위와 같은 사례에서도 알 수 있듯이, 오탐지를 야기하는 원인에 대하여 연관된 공통 코드 문맥 수집하여 분류/파악한다면, 검출 규칙을 편리하게 개선할 수 있음을 알 수 있다. 즉, 공통된 문맥 정보가 규칙 수정에 기여할 수 있는 유의미한 정보로 발전되고, 이를 유용하게 사용할 수 있다는 것이다.

뿐만 아니라, 이와 같은 과정을 통해 여러 공통 문맥 중, 오탐지의 근거가 되는 문맥을 질문의 형식으로 정리할 수 있다. 표 2에서는 CompareObjectsWithEquals라는 규칙의 오탐지 결과를 제거할 수 있는 코드 문맥 정보를 질문 형식으로 보여준다.

표 2. 결함 검출 규칙 개선을 위한 코드 문맥 정보

규칙	유형	Questions for extracting context
Compare Objects WithEquals	(1)Object/List/ Exception/ Type type간의 비교	비교되는 항이 Object type인가?
		비교되는 항이 List type인가?
		비교되는 항이 Exception type인가?
		비교되는 항이 Type type인가?
	(2)Map에서의 key 값에 의한 data 비교	비교되는 항이 Map에서의 key 값과 data 값인가?
	(4)method name에 set/add/store/equals를 포함한 method 내에서 비교	Method Name에 set, add, store, equals를 포함되어 있는가?
	(5)비교하는 line 밑에 바로 return; 수행	비교하는 항 바로 아래 return;이 있는가?
	(3)비교되는 두 항이 정의됨에 있어 서로 연관이 있는 경우	비교항이 for-each 구문 안에 들어가 있는가?
	(6)비교항의 변수 이름에 old/new /current가 포함된 경우	비교하는 항의 두 변수 이름이 일정 접두사를 제외하고 완전히 같은가?

3. 버그 검출 도구 규칙 수정을 위한 문맥 정보 자동 수집

앞 절의 분석 결과를 바탕으로 정적 분석 도구 규칙 수정을 위한 문맥 정보 자동 수집 방법을 다음과 같이 설계하였다.

3.1 오탐지 의심 사례 수집

2.1과 같은 방식으로, GitHub와 같은 오픈소스 레포지토리를 통해 하나의 프로젝트의 가장 최근 커밋된 소스 코드와 1년 전 코드, 혹은 코드가 작성된 지 1년이 되지 않은 코드라면, 초기의 코드를 비교할 수 있도록 두 소스 코드를 Clone및 Checkout한다.

다음으로, 임의의 PMD 규칙을 선택하여 두 코드에 동일하게 적용하여 csv 파일로 저장한다. 그리고 적용 결과를 비교하여 오탐지 의심 사례를 분류하고 수집한다. 분류 및 수집의 자동화를 위해 두 가지 분류 방법을 설정하였다. 첫 번째 방법은 결함 탐지 결과에서 같은 파일 명과 같은 라인 위치를 보이는 경우 False Positive로 분류하는 것이다. 두 번째 방법은 파일 명이 같으나 다른 라인 위치를 보이는 경우에 사용한다. 이 경우에는 나타나있는 파일 디렉토리에 자동 접근하여 소스 파일을 비교하는 절차를 거친다. 이 과정에서 결함으로 탐지된 라인 위치에 접근하여 해당 내용이 포함된 method 이름과 사용된 변수 이름을 비교한다. 모두 같을 때, 이 경우 또한 False Positive 의심 사례로 분류한다. 이와 같은 과정을 거치면 의심 사례 수집이 자동으로 완료되고 코드 문맥을 추출할 수 있게 된다.

3.2 오탐지 의심 사례들의 공통 코드 문맥 자동 추출

공통 코드 문맥을 자동 추출하는 방법은 다음과 같다. 먼저 False Positive 의심 사례에 해당하는 여러 코드의 문맥을 모두 파악한다. 그 다음으로 동일한 코드 문맥이 얼마나 발생하였는지 발생 횟수를 높은 순으로 정렬한다. 횟수가 높은 순으로 정렬된 내용에서 규칙 별로 문맥을 추출하기 위해 사전에 정의한 질문을 적용하여 다시 질문에 해당하는 내용이 많은 순으로 재정렬한다. 이를 통해 추출된 문맥은 False Positive 야기하는 횟수가 많으며, False Positive를 야기하는 확실한 원인 순으로 정렬될 것이다. 이는 검출 규칙 수정에 유의미한 정보를 제공하는 일종의 규칙 개선 척도로 사용될 수 있다.

3.3 추출된 문맥을 이용 검출 도구 규칙 개선

규칙 개선에 척도로 추출된 문맥을 이용하면 PMD 규칙 개선을 더욱 수월하게 할 수 있다. 기존의 규칙이 설정되어 있는 소스파일에 접근하여 해당 문맥이 규칙에 어떻게 구현되어 있는지 확인하고, 추출된 문맥에 따라 구현되어 있는 코드를 수정해주면 검출 도구 규칙 개선이 보다 쉽게 가능하다. 만약 추출된 문맥에 해당하는 내용이 검출 도구에서 구현되어 있지 않은 경우라면, 추출된 문맥에 따라 내용을 추가하면 규칙 개선이 용이하게 이루어질 수 있다.

4. 관련 연구

Jung et al.[9]의 연구는 ANSI C 프로그램에 buffer-overrun 결함을 발견하는 Airac(Array Index Range Analyzer for C) 이라는 정적 분석 도구를 이용한 뒤 통계적 분석을 통해 오탐지를 걸러내고 진짜 결함을 발견한 것에 대해 우선순위를 부여

하는 방식을 사용했다. 그 결과 75% 정도의 오탐지를 걸러냈다.

Yoon et al.[10]의 연구는 Support Vector Machine(SVM)을 이용하여 모델을 학습하고 새로운 alarm을 분류하여 오탐지를 제거한다. 이들은 SPARROW라는 정적 분석 도구로 실험한 결과 37% 정도의 오탐지를 제거했다.

Lee et al.[11]의 연구는 오탐지들이 결함이 발생하는 코드부분의 특정 문자 패턴을 식별함으로서 분류할 수 있다고 가정했다. 그리고 CNN을 바탕으로 만든 분류기로 문자 패턴을 포착할 수 있도록 하였다. 그 결과 79% 정도의 오탐지 결과를 제거할 수 있었다.

위와 같은 연구들이 통계적 기법을 사용한 것과 달리 본 연구는 마이닝 소프트웨어 레포지토리를 통해 기존 개발자들의 소스코드 수정 히스토리를 이용하여 오탐지 문제를 개선하고자 하는 방법을 고안했다.

5. 결론 및 향후 계획

본 연구는 정적 분석 도구가 지닌 다수의 오탐지 결과를 감소시키기 위해 정적 분석 도구의 설정된 규칙을 개선하는 과정의 효율성을 증대하는 것을 목표로 하였다. 현재까지도 정적 분석 도구에서 발생하는 많은 False Positive는 개발자의 혼란을 야기하고 개발자의 작업 비효율성을 높이는 문제가 있다. 하지만 본 연구에서 발견한 문맥 자동 수집 통한 정적 분석 도구의 규칙 개선은 이러한 문제점을 해결할 수 있다.

목표에 따른 실제 절차는 다음과 같다. 문맥 자동 수집 절차는 한 프로젝트의 최신 버전 소스코드와 1년 전 소스코드를 GitHub와 같은 오픈 소스 레포지토리에서 clone 및 checkout 하고 임의의 PMD 규칙을 적용한 뒤, csv 형식의 파일로 저장한다. 그 다음, 두 소스코드에 적용한 결과를 비교하여 False Positive 의심 사례를 수집하고 의심 사례에 나타난 문맥 파악 후, 문맥을 추출하기 위한 질문과 발생 빈도에 따라 정렬하여 규칙 수정에 유의미한 문맥을 수집한다. 이와 같은 정보를 통해 보다 쉽게 규칙 수정을 할 수 있고 추출된 문맥을 이용하여 개선한 규칙은 오탐지 결과를 감소시킬 것이다.

향후 연구는 고안한 문맥 자동 수집 절차를 이용하여 실제로 자동 수집이 가능한 도구인 문맥 추출기(Context Extractor)를 구현하는 것이다. 문맥을 기준으로 오탐지 결과의 원인이 되는 문맥을 자동 분류하여 규칙 수정에 활용한다면, 기존 정적 분석 도구의 높은 오탐지 비율 문제를 해결하는데 의미 있는 공헌을 할 수 있으리라 기대한다. 또한 관련 연구와의 오탐지 제거 비율을 지표로 한 정량적인 비교가 가능해지기 때문에, 기술의 실용성과 효율성에 대해 평가할 수 있을 것이라 기대된다. 더 나아가 자동으로 버그를 수정하는 최근 기술들을 접목시킨다면, 정적 분석 도구의 규칙도 자동으로 개선하는 도구 개발도 가능하리라 본다[12].

※ 본 연구는 과학기술정보통신부와 정보통신기술 진흥센터의 2019년도 소프트웨어 중심대학 지원사업(2017-0-00130)과 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2018R1C1B6001919).

6. 참고 문헌

[1] 도상혁, “한국IDC, 국내 상용 소프트웨어 시장 2022년 5조 2천억 전망”, Retrieved from: https://www.idc.com/getdoc.jsp?containerId=prAP44184618

[2] R. C. Seacord, D. Plackosh, and G. A. Lewis, Modernizing legacy systems: software technologies, engineering processes, and business practices. Addison-Wesley Professional, 2003

[3] PMD, “About PMD”, Retrieved from: https://pmd.github.io/

[4] WiseStone, “정적 분석도구 - SPARROW”, Retrieved from: http://www.wisestone.kr/solution/sparrow.php

[5] SonarQube, “SonarQube Continuous Inspection”, Retrieved from: https://www.sonarqube.org/

[6] Kim, Sunghun and Michael D. Ernst, “Prioritizing Warning Categories by Analyzing Software History.”, Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007), pp. 27, 2007.

[7] PMD, “Error Prone”, Retrieved from: https://pmd.github.io/pmd-6.10.0/pmd_rules_java_errorprone.html

[8] J. Nam, S. Wang, Y. Xi, L. Tan, “Designing Bug Detection Rules for Fewer False Alarms.” ICSE ‘18 Proceeding of the 40th International Conference on Software Engineering: Companion Proceedings , 2018.

[9] Jung Y., Kim J., Shin J., Yi K, “Taming False Alarms from a Domain-Unaware C Analyzer by a Bayesian Statistical Post Analysis”, Siveroni I. (eds) Static Analysis SAS 2005. Lecture Notes in Computer Science, vol 3672, 2005.

[10] J. Yoon, M. Jin and Y. Jung, "Reducing False Alarms from an Industrial-Strength Static Analyzer by SVM," 2014 21st Asia-Pacific Software Engineering Conference, pp. 3-6, 2014.

[11] Lee S., Hong S., Yi J., Kim T., Kim C., Yoo Shin, “Classifying False Positive Static Checker Alarms In Continuous integration Using Convolutional Neural Networks”, 2019 IEEE 12th International Conference on Software Testing, Verification and Validation (ICST), 2019.

[12] D. Kim, J. Nam, J. Song, S. Kim, “Automatic Patch Generation Learned from Human-written Patches”, in Proceedings of 35th International Conference on Software Engineering (ICSE 2013), pp. 77-80, 2013.

교차 프로젝트 기반 토픽 모델링과 네이브 베이지안 다항분포을 이용한 버그 리포트의 우선순위 예측

양근석 [1], 김재은 ○ [1], 이정원 [2], 이병정 [1]

[1] 서울시립대학교 컴퓨터과학과

{ypats87, jennyitem1902, bjlee}@uos.ac.kr

[2]아주대학교 전자공학과

jungwony@ajou.ac.kr

Bug Priority Prediction by using Cross Projects-based Topic Modeling and Naïve Bayes Multinomial

Geunseok Yang[1], Jaeeun Kim ○ [1], Jung-Won Lee[2], Byungjeong Lee[1]

[1]Department of Computer Science and Engineering, University of Seoul.

[2]Department of Electrical and Computer Engineering, Ajou University.

요 약

최근 소프트웨어의 복잡성과 규모는 점점 늘어나고 있다. 이에 따라 크고 작은 소프트웨어 버그를 피할 수 없게 되었다. 개발자들의 업무량과 소프트웨어 유지보수의 비용 또한 증가하고 있다. 소프트웨어 버그 정정 프로세스에서, 개발자는 본인이 할당 받은 소프트웨어 버그를 우선순위에 의해 정정 한다. 보통은 개발자들이 버그 리포트의 우선순위 속성을 변경한다. 이 과정에서 개발자들의 주관적인 판단과 노력이 반영될 수 있다. 만약 자동화된 버그 리포트 우선순위 예측 도구가 존재한다면, 개발자들의 업무량과 소프트웨어 유지보수 비용을 줄일 수 있다. 이를 해결하기 위해 본 연구에서는 교차 프로젝트 기반 토픽 모델링과 네이브 베이지안 다항분포 (Naïve Bayes Multinomial)를 이용하여 버그 리포트의 우선순위를 예측한다. 버그 리포트의 우선순위 예측 성능을 비교하기 위해 과거 전통적인 머신러닝 기법 (Naïve Bayes Multinomial, Naïve Bayes)을 Eclipse, Mozilla, GNU 오픈 소스 프로젝트에서 비교하였으며, 제안한 방법이 효율적임을 보인다.

1. 서 론

최근 소프트웨어는 다양한 산업에 적용되고 있다. 소프트웨어 규모와 복잡성 또한 계속 증가하고 있다. 이에 따라 크고 작은 소프트웨어 버그를 피할 수 없게 되었으며, Eclipse 오픈 소스 프로젝트에서는 하루에 약 300건의 버그 리포트가 제출되고 있다 [1]. 이러한 소프트웨어 버그 리포트에 따라, 개발자들의 업무량과 소프트웨어 유지보수의 비용 또한 증가하고 있다.

만약 소프트웨어 버그가 제출이 되면, 이를 할당 받은 개발자들은 일정에 맞추어 버그 심각성에 따라 작업의 우선순위를 정할 수 있다. 보통은 버그 추적 시스템 (Bug Tracking System)[2]에서 버그 리포트의 우선순위 속성을 제공한다. Bugzilla [3]에서는 버그 리포트의 우선순위가 P1 (우선순위 낮음) ~ P5 (우선순위 높음)으로 표현될 수 있고, 이 우선순위 속성을 개발자가 주관적인 판단으로 변경할 수 있다. 하지만, 하루에 많은 버그를 정정하는 개발자들에게는, 버그 리포트의 우선순위를 수동적으로 결정하는 데 많은 노력이 요구된다. 만약 자동화가 되면, 개발자들의 업무량과 소프트웨어 유지보수의 비용을 줄일 수 있다.

이를 해결하기 위해 본 논문에서는 교차 프로젝트 기반 토픽 모델링 [4]과 네이브 베이지안 다항분포 (Naïve Bayes Multinomial)[5]를 이용하여 버그 리포트의 우선순위를 예측한다. 즉, 과거 버그 리포트를 기반으로 하여 새롭게 들어오는 버그 리포트의 우선순위를 예측한다.

본 연구의 기여도 (Contribution)는 다음과 같다.

- 하루에 많은 버그를 정정해야하는 개발자들은 버그 우선순위를 아직도 수동적으로 정하고 있다. 여전히 많은 시간이 필요하며, 주관적인 판단이 반영될 수 있다. 만약, 자동화된 버그 리포트의 우선순위 예측 도구가 존재한다면, 개발자들의 업무량과 소프트웨어 유지보수

비용을 줄일 수 있다.

- 교차 프로젝트 기반 토픽 모델링과 Naïve Bayes Multinomial 기법을 사용하여 차별화된 모델을 만들고, 버그 우선순위의 예측 성능을 비교하기 위해 전통적인 머신러닝 기법 (Naïve Bayes Multinomial, Naïve Bayes)과 Eclipse, Mozilla, GNU 오픈 소스 프로젝트에서 비교하였으며, 제안한 방법이 효과적임을 보인다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 본 논문에서 필요한 배경 지식에 대해서 소개하고, 3장에서는 제안한 버그 리포트의 우선순위 예측 방법론을 기술한다. 4장에서는 제안한 방법론에 대해 실험을 진행하고 결과를 보인다. 5장에서는 실험 결과를 분석하여, 토의를 기술하고, 6장에서는 관련 연구를 소개한다. 7장에서는 결론 및 향후 연구에 대해 기술한다.

2. 배경 지식

2.1 버그 리포트

소프트웨어 버그는 소프트웨어 최종 사용자 (End User), 개발자 (Developer), QA (Quality Assurance) 등에 의해 발견된다. 보통은 특정한 양식에 작성이 되며, 자유로운 형태로 발견한 소프트웨어 버그를 기술 한다 [1]. 실제 Eclipse 오픈 소스 프로젝트에 제출된 버그 리포트 (#70010)[6]를 예제로 살펴보면, 다양한 속성들로 구성되어 있다. 이러한 속성들은 버그 리포트를 제출한 사람 (Reported, "*Tim Ellison*"), 버그를 제출한 시간 (Reported, "*July, 14, 2004*"), 정정된 시간 (Modified, "*July, 15, 2004*"), 심각도 (Importance, "*major*"), 우선순위 (Importance, "*P3*") 등을 포함하고 있다. 본 연구에서는 우선순위 속성을 예측한다.

2.2 토픽 모델링

일반적으로 문서 (Document)들은 특정한 주제(토픽)으로 분류될 수 있다 [7]. 본 연구에서는 토픽 모델링 기법을 이용하여 수집한 버그 리포트들을 다양한 토픽으로 분류한다. 자세히는, 버그 리포트들에 있는 단어들의 상호 빈도 (Co-occur Frequency)를 기반한 단어들의 분포 (Distribution)를 구성한다 [7].

3. 버그 우선순위 예측 방법

본 연구는 교차 프로젝트 기반 토픽 모델링과 Naïve Bayes Multinomial을 이용하여 버그 우선순위를 예측한다. 전반적인 도식도는 그림 1와 같다.

먼저, 버그 리포트의 전처리 과정을 진행하고, 교차 프로젝트 기반 토픽 모델링을 구축한다. 그리고 Naïve Bayes Multinomial 기법을 통해 모델을 훈련하고, 최종적으로, 새로운 버그 리포트의 우선순위 예측을 한다.

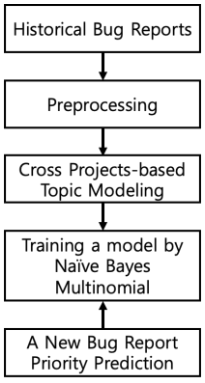


그림 1. 전반적인 도식도

3.1 버그 리포트 전처리

버그 리포트는 자유로운 문맥 (Free-form Textual Contexts)으로 기술되며 [1], 문장을 의미 있는 단어들로 분류하고, 분류된 단어들을 의미 있게 정제할 필요가 있다. 본 논문에서는 전처리 과정에서 문장의 토큰화 (Tokenization), 어원 추출 (Stemming), 정지 단어 제거 (Stop words Removal)을 진행한다.

3.2 교차 프로젝트 기반 토픽 모델링

토픽 모델링을 형성하기 위해, 저장된 버그 리포트의 요약 (Summary)과 설명 (Description)를 사용한다. 자세히는 Eclipse [8], Mozilla [9], GNU [10]의 버그 리포트를 모두 활용하여 토픽 모델을 구축한다. 구축된 각 토픽들은 다양한 단어들로 구성되어 있으며, 이 단어들의 분포 점수들은 다양하게 표현된다. 단어들의 분포 점수가 높을 수록 해당 단어가 해당 토픽에 연관되어 있다고 볼 수 있다.

본 논문에서는 새로운 버그 리포트가 도착했을 때, 전처리 과정을 거치고, 해당 단어들로 토픽들의 단어들과 비교하여, 가장 유사한 토픽을 찾는다.

3.3 버그 리포트의 우선순위 예측 기법

새로운 버그 리포트로부터 유사한 토픽을 찾고, 해당 토픽 내 버그 리포트들을 추출한다. 추출된 버그 리포트를 이용하여 Naïve Bayes Multinomial 기법 [5]에 모델을 훈련 시킨다. 일반적으로 Naïve Bayes Multinomial 기법은 단어들의 존재 유무, 발생 빈도를

사용하여, 문서 내 구분되는 단어들이 많을 수록 Naïve Bayes 기법[5] 보다 좋은 성능을 보일 수 있다 [5].

본 논문에서는 훈련된 Naïve Bayes Multinomial 모델을 사용하여 새로운 버그 리포트의 우선순위를 예측한다.

4. 실험

본 논문에서는 Eclipse [8], Mozilla [9], GNU [10] 오픈소스 프로젝트로부터 버그 리포트를 수집하였다. 사용된 버그 리포트는 표 1과 같다.

표 1. 사용된 버그 리포트

Project	Count	기간
Eclipse	1,000	'02/01/01 ~ '05/06/14
Mozilla	1,000	'02/09/30 ~ '05/12/30
GNU	1,000	'00/02/26 ~ '03/09/15

4.1 실험 요약

수집된 버그 리포트를 교차 프로젝트 기반 토픽 모델링 기법을 사용하여 토픽을 구성한다. 토픽 모델 구축에서 Stanford Topic Modeling Toolbox [11] 도구를 사용하고, 파라미터는 α=0.01, β=0.01, γ=1500, n=30으로 사용한다. α와 β를 조절하면, 토픽 내 속할 버그 리포트들과 단어들의 비율을 조절할 수 있다. γ는 토픽을 구성할 때, 훈련되는 횟수이고, n은 30개의 토픽을 구성한다 [7]. 새로운 버그 리포트에 전처리 기법을 사용하고, Stanford coreNLP [12] 도구를 사용한다. 이 과정에서, 문장으로부터 의미 있는 단어들을 추출하고, 구축된 토픽들로부터 유사한 토픽을 찾는다. 유사한 토픽 내, 버그 리포트들을 추출하고 Naïve Bayes Multinomial 기법을 사용하여 모델 훈련한다. 그리고, 새로운 버그 리포트의 우선순위를 예측한다. 제안한 방법은 교차 프로젝트 기반 토픽 모델링 기법을 사용하였으므로, 공정한 비교를 위해, 전통적인 머신러닝 기법 (Naïve Bayes Multinomial, Naïve Bayes)들도 각각 Eclipse, Mozilla, GNU의 결과를 평균 내어 비교를 진행한다.

4.2 실험 평가척도

본 논문에서는 구축한 모델의 성능을 평가하기 위해 일반적으로 사용되는 Precision [13], Recall [13], F-Measure [13] 척도를 사용한다. 추가적으로 데이터의 편향 (bias)을 줄이기 위해 10-Fold 교차 검증 (Cross Validation)[14]을 진행한다. 공정한 실험을 위해 전통적인 머신러닝 기법 (Naïve Bayes Multinomial, Naïve Bayes)에도 적용한다.

4.3 실험 결과

RQ. 버그 리포트의 우선순위를 예측하기 위해 어떠한 기법이 가장 효율적인가?

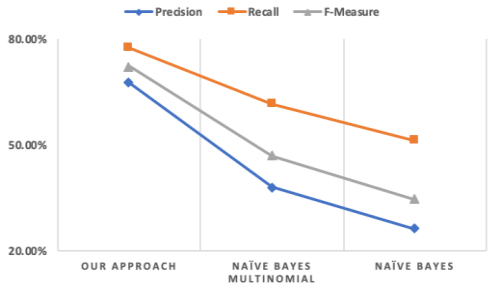


그림 2. 다른 방법과 비교 평가

그림 2에서, X축은 제안한 기법과, 비교를 위한 전통적인 머신러닝 기법 (Naïve Bayes Multinomial, Naïve Bayes)을 의미하고, Y축은 10-Fold 교차 검증의 평균 척도 값 (Precision, Recall, F-Measure)을 의미한다. 제안한 기법이 더 잘 예측하는 것으로 보인다.

5. 토의

5.1 실험 결과

본 논문은 교차 프로젝트 기반 토픽 모델링과 Naïve Bayes Multinomial 모델을 사용하여 버그 리포트의 우선순위를 예측하였다. Naïve Bayes Multinomial 기법의 특성을 이용하여 다양한 단어들의 존재 유무 및 발생 빈도를 반영하였다. 따라서, 전통적인 기법 (Naïve Bayes, Naïve Bayes Multinomial) 보다 버그 우선순위 예측 성능을 더 향상할 수 있었다.

5.2 위협 요소

버그 리포트의 우선순위를 예측하기 위해 Eclipse, Mozilla, GNU 오픈 소스 프로젝트로부터 버그 리포트를 수집하였다. 하지만, 제안한 기법이 비즈니스 프로젝트에서도 효율적인 성능을 보인다고 주장할 수 없다. 일반적으로 오픈 소스 프로젝트와 비즈니스 프로젝트에서의 버그 정정 관리 프로세스는 다르기 때문이다 [1]. 그리고, Eclipse, Mozilla, GNU의 버그 리포트를 활용한 것이 항상 좋은 성능을 보인다고 주장할 수 없다. 이에 대해 더 많은 데이터 셋과 다양한 요소들을 활용한 추가 연구가 필요하다. 본 논문에서는 다른 (프로젝트) 버그 리포트의 우선순위가 높을 때, 현재 프로젝트에서 높을 것이라는 가정으로 접근하였다. 하지만, 프로젝트 마다 성격과 프로세스가 다르므로,

이에 대해 추가 연구가 필요하다. 향후 더 많은 데이터 셋과 교차 검증 뿐만 아니라, 적당한 비율 (Train/Test)을 통하여 제안한 논문의 효율성을 더 깊게 검증할 예정이다.

6. 관련 연구

Tian [15]은 Multi-Factor 분석을 통해 버그 리포트의 우선순위를 예측했다. 사용된 Multi Factors [15]는 temporal, textual, author, related-report, severity, product 이며, Eclipse 프로젝트에서 예측 성능이 효율적임을 보였다.

Sharma [16]는 머신 러닝 기법들과 교차 프로젝트를 사용하여 버그 리포트의 우선순위를 예측했다. Eclipse와 Open Office [16]를 사용하였으며, Information Gain [16] 기반 특징 추출 및 Rapid Miner [16]를 통해 효율적인 성능을 보였다.

하지만, 위 연구들 모두 버그 리포트의 우선순위 예측 성능이 향상될 필요가 있으며, 다양한 프로젝트에서도 추가적인 검증이 필요하다. 본 연구에서는 교차 프로젝트 (Eclipse, Mozilla, GNU) 기반 토픽 모델링과 Naïve Bayes Multinomial 기법을 사용하여, 과거 관련 연구들과 차별성을 보인다.

7. 결론

소프트웨어의 규모와 복잡성 증가로 크고 작은 버그가 존재하게 되었다. 이에 대해 개발자들의 업무량과 소프트웨어 유지보수의 비용 또한 증가하였다. 이를 해결하기 위해 본 논문에서는 교차 프로젝트 기반 토픽 모델링과 Naïve Bayes Multinomial 기법을 이용하여 모델을 구축했다. 제안한 모델의 성능을 평가하기 위해 전통적인 머신러닝 기법 (Naïve Bayes Multinomial, Naïve Bayes)들과 Eclipse, Mozilla, GNU 오픈 소스 프로젝트에서 비교하였으며, 제안한 기법이 더 효율적임을 보였다. 향후 더 많은 버그 리포트와 다양한 요소들, 기법들을 활용하여 성능을 더 향상할 예정이다.

7. Acknowledgement

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 차세대정보·컴퓨팅기술개발사업(NRF-2014M3C4A7030504)과 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 중견연구자지원사업(No.2017R1A2B4009937)의 지원을 받아 수행된 연구임.

8. 참고 문헌

[1] G. Yang, T. Zhang, and B. Lee, "An Emotion Similarity based Severity Prediction of Software Bugs: A Case Study of Open Source Projects," IEICE TRANSACTIONS on Information and Systems, Vol.E101-D, No. 8, pp. 2015-2026, Aug. 2018.

[2] N. Jalbert and W. Weimer, "Automated Duplicate Detection for Bug Tracking Systems," In Proc. of IEEE International Conference on Dependable Systems & Networks, pp. 52-61, 2008.

[3] Bugzilla, https://www.bugzilla.org/.

[4] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, Vol. 3, pp. 993-1022, 2003.

[5] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug", In Proc. of Software Maintenance and Reengineering, pp. 249-258, 2011.

[6] Eclipse Bug Report #70010, https://bugs.eclipse.org/bugs/show_bug.cgi?id=70010.

[7] G. Yang, T. Zhang, and B. Lee, "Towards Semi-Automatic Bug Triage and Severity Prediction Based on Topic Model and Multifeature of Bug Reports," In Proc. of Computer Software and Applications Conference, pp.97-106, 2014.

[8] Eclipse, https://bugs.eclipse.org/bugs/.

[9] Mozilla, https://bugzilla.mozilla.org/.

[10] GNU, https://debbugs.gnu.org/.

[11] Stanford Topic Modeling Toolbox, https://nlp.stanford.edu/software/tmt/tmt-0.4/.

[12] C. D. Manning, M. Surdeanu, J.Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit", In Proc. ACL (System Demonstrations), pp. 55-60, 2014.

[13] C. Goutte and E. Gaussier, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation," Lecture Notes in Computer Science, Vol. 3408, pp. 345-359, 2005.

[14] R. Kohavi. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," Ijcai. Vol. 14. No. 2. 1995.

[15] Y. Tian, D. Lo, and C. Sun, "Drone: Predicting Priority of Reported Bugs by Multi-Factor Analysis," In Proc. of IEEE International Conference on Software Maintenance, pp. 200-209, 2013.

[16] M. Sharma, P. Bedi, K.K. Chaturvedi, and V.B. Singh, "Predicting the Priority of a Reported Bug using Machine Learning Techniques and Cross Project Validation," In Proc. of International Conference on Intelligent Systems Design and Applications (ISDA), pp. 539-545, 2012.

안전 필수 시스템의 ISO/IEC 16085기반 위험 분석 방법 비교

이현호 ○ 최현재 채흥석
부산대학교 전기전자컴퓨터공학과
lhh8048@pusan.ac.kr, gonoki@pusan.ac.kr, hschae@pusan.ac.kr

Comparison of risk analysis based on ISO / IEC 16085 for safety-critical systems

Hyeon-Ho Lee○ Hyun-Jae Choi Heung-Seok Chae
Dept. of Electrical & Computer Engineering, Pusan National University

요 약

안전 필수 시스템은 시스템의 오동작이 인간의 생명, 재산, 환경에 막대한 피해를 줄 수 있는 시스템이다. 안전 필수 시스템의 오동작은 치명적인 인명, 경제, 환경 피해를 야기할 수 있으므로 위험 분석을 통해 오동작으로 인한 피해를 최소화할 필요가 있다. 본 논문에서는 5개의 안전성 필수 시스템 위험 분석 논문을 선정하여, 선정된 논문을 위험 관리 표준인 ISO/IEC 16085를 기반으로 비교한다. 위험 분석의 세부 절차인 위험 식별, 위험 산정, 위험 평가 각각에 대해 각 논문의 수행 방법을 소개하고 수행 방법 별 유사점, 차이점을 설명한다.

1. 서 론

안전 필수 시스템은 시스템의 오동작이 인간의 생명 위협, 대량의 경제적 손실, 광범위한 환경 피해를 초래할 수 있는 시스템이다[1]. 안전 필수 시스템에서의 오동작은 치명적인 인명, 경제, 환경 피해를 야기할 수 있으므로 시스템의 오동작으로 인한 피해를 줄이기 위해 위험 통제 및 최소화를 수행할 필요가 있다.

시스템의 크기와 복잡도가 증가함에 따라 위험의 통제 및 최소화의 필요성이 더욱 대두되었다[2]. 규모가 크고 복잡도가 높은 시스템의 위험 통제 및 최소화를 위해서는 체계적인 위험 분석 절차가 요구된다. 본 논문에서는 체계적인 위험 분석 절차를 설명하고 위험 분석 절차의 각 단계별 수행 방안을 소개한다.

본 논문에서는 위험 관리 표준인 ISO/IEC 16085를 기반으로 위험 분석의 3단계 절차인 위험 식별, 위험 산정, 위험 평가를 설명한다. 그리고 안전 필수 시스템의 위험 분석 방법에 대한 논문을 선정하고 선정 논문에 대해 간략히 설명한다. 대상 논문들의 분석 기준을 제시하고 분석 기준에 따라 대상 논문들의 방법을 설명하고 방법 별 유사점, 차이점을 분석한다.

2. 위험 분석 절차

위험 분석은 위험 관리에서 프로젝트의 위험 관련 정보를 업데이트하고 유지하는 과정이다. 그리고 위험 분석은 위험 식별, 위험 산정, 위험 평가의 3 단계의 절차를 가진다[3]. 그림 1은 위험 분석의 3단계와 각 단계의 산출물을 표현한다.

위험 식별은 프로젝트에 있는 위험을 식별하는 과정이다. 위험 식별은 위험에 연관된 이벤트와 위험이 일어나는 근원 요소를 찾는 단계이며 접근법에는 설문, 위험 카테고리 분류, 브레인 스토밍, 시나리오 분석, 교훈 등이 사용된다.

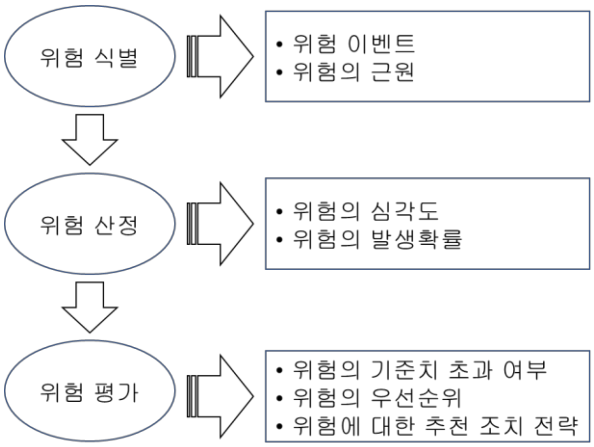


그림 1 위험 분석 절차 및 산출물

위험 산정은 위험의 발생 확률과 그 결과를 산정하는 과정이다. 위험 산정은 정성적 혹은 정량적으로 위험의 발생 확률과 그 결과의 심각도를 측정한다.

위험 평가는 식별, 산정된 위험에 대해 평가하는 과정이다. 위험 평가를 통해 해당 위험의 프로젝트에서 정한 위험 기준치 초과 여부를 평가하고 각 위험 간의 우선순위를 결정한다. 또한 위험에 대한 추천 조치 전략도 고려된다.

3. 대상 논문 선정

3.1 대상 논문 선정 기준

대상 논문은 안전 필수 시스템의 위험 분석 방법에 대한 논문으로 한다. 위험 분석의 3단계인 위험 식별, 위험 산정, 위험 평가를 가지고 있으며, Risk Management 및 Safety Risk Assessment관련 논문 중 게재 년이 2012년도 이후이고 인용 횟수 10이상인 논문을 비교 대상 논문으로 선정하였다.

3.2 선정 논문 요약

Ahn and Chang [4]은 위험 평가 값의 불확실성을 줄이는 방법을 제시하는 논문이다. 시스템의 위험 상세 분류 매트릭스를 만들어 위험 평가를 개선하였고 LNG 관련장비 시스템에 대하여 사례 연구를 수행하였다.

Khalil et al. [5]은 안전 무결성 수준(SIL)을 평가하는 개선된 방법을 제안하는 논문이다. 위험에 대비해 필요한 장비의 등급인 안전 무결성 기준을 이용해 위험 평가 과정을 수행하였고 가스 플랜트 시스템에 대하여 사례 연구를 수행하였다.

Shariff and Zaini [6]는 예비 설계 단계에서 위험 분석하는 방법을 제시하는 논문이다. 프로세스 디자인 시뮬레이터를 이용한 사고 시나리오 분석으로 위험 식별 과정을 수행하였고 암모니아 생산 공장 시스템에 대하여 사례 연구를 수행하였다.

Yazdi [7]는 전문가 집단의 정성적인 위험 산정 값을 통합하여 위험의 우선순위를 정하는 방법을 제시하는 논문이다. 시스템의 위험의 심각도와 발생 확률 계산의 새로운 방법을 제안하여 위험 산정 과정을 개선하였고 가스 정제소 관련 시스템에 대하여 사례 연구를 수행하였다.

Aminbakhsh et al. [8]은 시스템의 위험 카테고리 간의 우선 순위 결정 방법을 제안하는 논문이다. 위험의 각 카테고리들 간의 비교를 통한 각 위험의 심각도를 계산하는 위험 산정 과정을 수행하였고 건설 프로젝트에 대하여 사례 연구를 수행하였다.

4. 분석 기준

논문들이 시행한 방법들을 위험 분석의 3단계인 위험 식별, 위험 산정, 위험 평가로 나눈다. 각 위험 분석 단계에서 사용한 위험 분석 방법과 각 방법의 특징을 분석한다.

대상 논문들의 위험 식별 방법을 위험 식별 기반 정보와 위험 식별 대상 수준을 기준으로 비교한다. 위험 식별 기반 정보는 시스템 설계 정보와 과거 위험사례 정보이고 위험 식별 대상 수준은 시스템 파라미터, 시스템 기능, 위험 시나리오, 위험 요소, 위험 카테고리이다.

대상 논문들의 위험 산정 방법을 위험 산정 기반 정보와 산정 대상 위험을 기준으로 비교한다. 위험 산정 기반 정보는 위험 산정표와 전문가 평가이고 위험 산정 대상 위험은 시스템의 단일 위험, 위험 요소, 위험 카테고리이다.

대상 논문들의 위험 평가 방법을 위험 평가 결과와 평가 대상 위험으로 비교한다. 위험 평가 결과는 위험 수준 분류와 위험 우선 순위이고 평가 대상 위험은 시스템의 단일 위험, 위험 요소, 위험 카테고리이다.

5. 분석 결과

5.1 위험 식별 접근법 비교

대상 논문들이 사용한 위험 식별 방법을 분석하여 위험식별에 필요한 기반 정보를 기준으로 시스템 설계 정보와 과거 위험사례 정보로 나눈다. 또한 각 대상 논문들의 위험 식별 방법의 위험 식별 대상 수준을 비교한다. 표 1은 대상 논문들을 위험 식별 방법을 위험 식별 기반 정보와 위험의 식별 대상 수준에 대해 비교한 결과이다.

표 1 위험 식별 접근법 비교

대상 논문	위험 식별 방법	식별 기반정보	식별 대상 수준
Ahn and Chang [4]	HAZOP	시스템 설계정보	시스템 파라미터
Khalil et al. [5]	PHA		시스템 기능
Shariff and Zaini [6]	프로세스 설계 시뮬레이터	과거 위험사례 정보	위험 시나리오
Yazdi [7]	전문가 경험		위험 요소
Aminbakhsh et al. [8]	카테고리 분류		위험 카테고리

위험 식별은 시스템 설계 정보와 과거 위험 사례 정보를 기반으로 수행되었다. Ahn and Chang와 Khalil et al.은 시스템의 설계, 프로세스 데이터와 같은 설계 정보를 입력 받아 체계적인 분석을 통해 위험을 식별하였다. Shariff and Zaini, Yazdi 그리고 Aminbakhsh et al.는 유사 시스템의 위험 시나리오, 위험 관련 문헌과 같은 과거 위험 사례 정보를 기반으로 위험을 식별하였다.

Ahn and Chang은 HAZOP 방법[9]을 이용하여 시스템의 각 기능이 사용하는 시스템 파라미터와 각 기능에 가이드 워드를 주입함으로써 기능의 비정상적인 시나리오를 생성해 위험을 식별한다. 이와 달리 Khalil et al.은 PHA 방법[9]을 이용하여 시스템의 각 기능을 대상으로 체크리스트를 적용해 각 기능의 발생 가능 위험을 식별한다.

Shariff and Zaini는 프로세스 설계 시뮬레이터를 이용하여 위험 식별을 수행한다. 이 방법은 시스템의 초기 설계 정보를 이용해 생성한 프로세스 설계 시뮬레이터에 과거 유사 시스템의 사고 시나리오를 주입하여 위험을 위험 시나리오 단위로 식별한다. 반면 Yazdi는 유사 시스템에 대한 전문가들의 경험, 경력, 교육 수준을 기반으로 위험을 용접 빔 노출, 화학 물질 노출, 전기 충격과 같은 위험 요소 형태로 식별하였다. Aminbakhsh et al.는 과거 유사 시스템의 사례를 참고하여 물리적 위험, 화학적 위험과 같은 대상 시스템의 발생 가능 위험들을 카테고리 형태로 분류하여 위험을 식별하였다.

5.2 위험 산정 방법 비교

대상 논문들이 사용한 위험 산정 방법을 분석하여 위험 산정 기반 정보를 기준으로 비교 산정표 기반 방법과 전문가 평가 기반 방법으로 나눈다. 또한 각 대상 논문이 사용한 위험 산정 방법의 산정 대상 위험에 따른 특징을 비교한다. 표 2는 대상 논문들을 위험 산정 방법을 위험 산정 기반 정보와 산정 대상 위험에 대해 비교한 결과이다.

표 2 위험 산정 방법 비교

대상 논문	위험 산정 방법	위험 산정 기반 정보	산정 대상 위험
Ahn and Chang [4]	Frequency Consequence	위험 산정표	시스템의 단일 위험
Khalil et al. [5]	Safety, Frequency Economical effect		
Shariff and Zaini [6]	Severity Likelihood		
Yazdi [7]	퍼지 기반 소속도 가중산술평균	전문가 평가	위험 요소
Aminbakhsh et al. [8]	상대점수기반 단순산술평균		위험 카테고리

위험 산정은 위험 산정표와 전문가 평가 기반으로 수행되었다. Ahn and Chang, Khalil et al. 그리고 Shariff and Zaini은 객관적인 산정 기준표인 위험 산정 표를 만들어 위험을 산정하였다. Yazdi 와 Aminbakhsh et al.는 위험에 대한 전문가의 평가 후 각 평가를 통합하여 위험을 산정하였다.

Ahn and Chang 은 미 국방부 시스템 안전 표준 MIL-STD-882E[10]을 참고해 Frequency 와 Consequence 에 대한 위험 산정 표를 만들어 시스템의 단일 위험에 대해 위험 산정 과정을 수행한다. 반면 Khalil et al.은 사람에 대한 위험의 심각도인 Safety 와 위험의 경제적 심각도인 Economical effect 를 이용하여 위험의 심각도인 severity 에 대한 위험 산정표를 만든다. 그리고 관련 문헌을 참고해 위험의 Frequency 에 대한 위험 산정표도 만들어 시스템의 단위 위험에 대해 위험 산정 과정을 수행한다. Shariff and Zaini 는 미 국방부 시스템 안전 표준 MIL-STD-882D[11]을 참고해 위험의 likelihood 에 대한 위험 산정표를 만든다. 그리고 화학 물질 노출에 대한 위험 농도 기준인 AEGLs[12]을 참고해 위험의 Severity 에 대한 위험 산정표도 만들어 시스템의 단일 위험에 대해 위험 산정 과정을 수행한다.

Yazdi 는 퍼지 기반 소속도 가중산술평균 방법을 사용하여 위험 산정 과정을 수행한다. 전문가들이 각 위험의 심각도와 발생 확률을 언어적 표현으로 산정하고 평가 결과를 퍼지 기반의 소속도 값으로 변환한다. 각 위험에 대한 전문가의 산정 값을 가중산술평균으로 통합하는 과정을 통해 위험 요소를 대상으로 위험 산정 과정을 수행한다. 반면 Aminbakhsh et al.는 상대점수기반의 단순산술평균으로 위험 산정 과정을 수행한다. 각 위험 카테고리 간 중요도 비교 값을 상대 값으로 측정 후 변환하여 각 위험의 심각도을 산정한다. 또한 전문가 평가로 각 위험의 발생 확률을 산정하고 단순 산술 평균으로 통합하여 위험 카테고리 대상의 위험 평가 과정을 수행한다.

5.3 위험 평가 방법 비교

대상 논문들의 위험 평가 방법을 분석하고 위험 평가 결과를 기준으로 위험 수준 분류와 위험 우선 순위 결정 방법으로 나눈다. 또한 각 대상 논문이 사용한 위험 평가 방법의 평가 대상 위험에 따른 특징을 비교한다. 표 3은 대상논문들의 위험 평가 방법을 위험 평가 결과와 평가 대상 위험에 대해 비교한 결과이다.

표 3 위험 평가 방법 비교

대상 논문	위험 평가 방법	위험 평가 결과	평가 대상 위험
Ahn and Chang [4]	퍼지 Risk Matrix	위험 수준 분류	시스템의 단일 위험
Khalil et al. [5]	SIL 평가		
Shariff and Zaini [6]	Risk Matrix		
Yazdi [7]	IFH-TOPSIS	위험 우선 순위	위험 요소
Aminbakhsh et al. [8]	단순비교연산		위험 카테고리

위험 평가는 위험 수준 분류와 위험 우선 순위로 수행되었다. Ahn and Chang, Khalil et al. 그리고 Shariff and Zaini은 각 위험에 대한 위험 수준 분류기준을 만들어 그에 따라 위험을 분류하는 방법으로 위험을 평가하였다. Yazdi 와 Aminbakhsh et al.는 위험들 간의 비교를 통해 위험에 대처하는 우선 순위를 설정하는 방법으로 위험을 평가하였다.

Ahn and Chang은 표의 영역을 통해 위험 수준을 분류하는 방법인 퍼지 Risk Matrix 방법을 사용한다. Risk Matrix는 위험의 심각도와 발생 확률을 매핑하여 해당 위험의 위험 수준을 분류하는 방법이며 분류 경계를 단계적으로 나누는 퍼지 이론[13]을 적용해 시스템의 단일 위험에 대한 더 세부적인 위험 수준 분류를 시도하였다. 반면 Khalil et al.은 안전을 위해 현장에 설치될 계기설비의 등급을 나타내는 안전 무결성 수준(SIL) 평가[14]을 통해 각 시스템 단일 위험의 조치 등급에 대한 위험 평가 과정을 수행한다. Shariff and Zaini는 일반적인 Risk Matrix을 통해 위험 평가 과정을 수행하지만 위험 평가 수준을 2단계로 하여 각 시스템 단일 위험의 조치 필요 여부만 판단한다.

Yazdi는 이 논문에서 제안하는 Fuzzy기반 소속도 값을 이용해 이상적인 솔루션과의 유사성을 측정하는 IFH-TOPSIS방법을 통해 위험을 평가한다. IFN-TOPSIS은 각 위험의 심각도와 발생확률에 가중치를 적용해 최악의 위험과 해당 위험의 유사성 값 계산하며 이를 기준으로 위험 요소들 간의 우선 순위를 결정하는 과정이다. 반면 Aminbakhsh et al. 은 위험 카테고리들 간의 비교 값인 위험의 심각도와 위험 발생 확률을 단순 곱 연산한 위험 수준 비교 값을 만들어 위험 카테고리들 간의 우선 순위를 정한다.

6. 결 론

본 논문에서는 안전 필수 시스템을 사례 연구로 하는 위험 분석 방법에 대한 논문을 선정해 비교해 보았다. ISO/IEC 16085 표준에 명시된 위험 분석의 절차를 분석 기준으로 정하고 대상 논문이 각 단계에서 수행한 방법을 비교해 보았다.

위험 식별 단계에서는 대상 논문들을 식별 기반 정보와 위험 식별 대상 수준을 기준으로 비교하였다. 식별 기반 정보는 시스템 설계 정보 기반 분석과 과거 사례 정보 기반 분석으로

분류하였으며 시스템 설계 정보 기반 방법에는 Ahn and Chang와 Khalil et al.가 분류되었고 과거 사례 정보 기반 분석에는 Shariff and Zaini, Yazdi 그리고 Aminbakhsh et al.가 분류되었다. 또한 각 논문의 식별 대상 수준을 비교한 결과 가장 세부적인 식별 대상은 Ahn and Chang이 사용한 HAZOP 방법이 대상으로 하는 시스템 기능 파라미터이다. 위험 산정 단계에서는 대상 논문들을 위험 산정 기반 정보와 산정 대상 위험을 기준으로 비교하였다. 위험 산정 기반 정보는 위험 산정표 기반과 전문가 평가 기반으로 분류하였으며 위험 산정표 기반은 Ahn and Chang, Khalil et al. 그리고 Shariff and Zaini이 분류되고 시스템의 단일 위험 대상을 위험 산정하였다. 전문가 평가 기반 분석에는 Yazdi와 Aminbakhsh et al.이 분류되었으며 위험 요소, 위험 카테고리 대상으로 위험 산정하였다. 위험 평가 단계에서는 대상 논문들을 위험 평가 결과와 위험 평가 대상을 기준으로 비교하였다. 위험 평가 결과는 위험 수준 분류와 위험 우선 순위로 비교하였으며 위험 수준 분류에는 Ahn and Chang, Khalil et al. 그리고 Shariff and Zaini이 분류되었고 시스템의 단일 위험을 평가 대상으로 하였다. 위험 우선 순위 방법에는 Yazdi와 Aminbakhsh et al.이 분류되었고 각각 위험 요소와 위험 카테고리를 위험 평가 대상으로 하였다.

참 고 문 헌

[1] J. C. Knight, "Safety critical systems: challenges and directions," *Proc. of the 24th International Conference on Software Engineering*, pp.547-550, 2002.

[2] J. L. Fuentes-Bargues, C. González-Gaya, M. C. González-Cruz, V. Cabrelles-Ramírez, "Risk assessment of a compound feed process based on HAZOP analysis and linguistic terms," *Journal of Loss Prevention in the Process Industries*, Vol. 44, pp. 44-52, November. 2016.

[3] "*ISO/IEC 16085 : Systems and software engineering - Life cycle processes - Risk management*," ISO, 2006.

[4] J. Ahn, D. Chang, "Fuzzy-based HAZOP study for process industry," *Journal of Hazardous Materials*, Vol. 317, pp. 303-311, November. 2016.

[5] M. Khalil, M. A. Abdou, M. S. Mansour, H. A. Farag, M. E. Ossman, "A cascaded fuzzy-LOPA risk assessment model applied in natural gas industry," *Journal of Loss Prevention in the Process Industries*, Vol. 25, pp. 877-882, November. 2012.

[6] A. M. Shariff, D. Zaini, "Inherent risk assessment methodology in preliminary design stage : A case study for toxic release," *Journal of Loss Prevention in the Process Industries*, Vol. 26, pp. 605-613, July. 2013.

[7] M. Yazdi, "Risk assessment based on novel intuitionistic fuzzy-hybrid-modified TOPSIS approach," *Safety Science*, Vol. 110, pp. 438-448, December. 2018.

[8] S. Aminbakhsh, M. Gunduz, R. Sonmez, "Safety risk assessment using analytic hierarchy process (AHP) during planning and budgeting of construction projects," *Journal of Safety Research*, Vol. 46, pp. 99-105, September. 2013.

[9] C. A. Ericson II, *Hazard Analysis Techniques for System Safety*, John Wiley & Sons, 2005

[10] Department of Defense, "*MIL-STD-882E : Department of Defense Standard Practice System Safety*," 2012

[11] Department of Defense, "*MIL-STD-882D : Department of Defense Standard Practice System Safety*," 2000

[12] National Research Council, *Standing Operating Procedures for Developing Acute Exposure Guideline Levels for Hazardous Chemicals*, National Academies Press, Washingtion, 2001.

[13] L. A. Zadeh, "Fuzzy sets," *Information and Control*, Vol. 8, pp. 338-353, June. 1965.

[14] "*IEC 61508 : Functional safety of electrical/electronic /programmable electronic safety-related systems*," IEC, 1998

자율 군집 주행에서 그룹이탈 경보시스템의 Wright 기반 명세

윤이진 ○, 남승우, 홍장의
충북대학교 소프트웨어학과
483759@naver.com, Jehong@chungbuk.ac.kr

Specifying seceding alarm system of autonomous platoon driving using Wright notations

Ijin Yun, Seungwoo Nam, Jang-Eui Hong
Dept. of Computer Science, Chungbuk National University

요 약

자율주행자동차는 센서 및 영상처리 장치 등을 이용하여 사람의 개입 없이 주행하는 자동차를 의미한다. 특히 자율주행 연구분야에 V2X 통신을 이용하여 차량간에 통신하고, 여러 차량이 군집을 형성하여 선두 차량이 통제권을 제어하는 군집주행 기술이 핵심 기술로 떠오르고 있다. 그러나 군집주행 시스템에서 원활한 주행을 유지하기 위해서는, 그룹을 이탈하는 차량과의 상호작용이 안정적으로 이루어지는 것이 중요하다. 따라서 본 연구에서는 군집주행 기술에서 V2V 통신을 활용한 그룹 이탈 경고 시스템에 대한 알고리즘을 제시하고, 이를 Wright 명세와 상태-전이 다이어그램으로 명세하여 제시된 행위가 올바르게 동작하는지를 확인할 수 있도록 지원하였다.

1. 서 론

자율주행 자동차의 소프트웨어(Autonomous Driving Software. 이하 ADS)는 자율주행 기술의 발전과 요구에 따라 중요성과 실현 가능성을 인정받아 지속적인 연구가 진행되어 오고 있다[1]. 2018년 구글(Google)은 400명의 제한된 사용자에게 우선적으로 자율 주행 택시의 실제 운행 서비스를 도입했다[2]. 또한 현대기업은 '2018 CES'에서, 오로라와의 기술 협력을 통해 2021년까지 레벨4의 자율주행자동차를 상용화 할 것이라 밝혔다[3].

다양한 자율주행 관련 기술이 연구되면서, 다수의 자동차가 그룹을 형성하여 일정한 대열로 자율 주행하는 군집주행에 대한 연구도 진행되고 있다. 군집주행 기술은 자동차간 통신이 요구되는 새로운 방식의 기술이 요구되며, 자율 주행의 효율성 등으로 인하여 최근 주목 받는 연구분야이다.

군집주행 기술 중에서 , 본 논문은 군집 주행에서의 이탈 과정에 대하여 연구 기술한다. 군집을 이루는 자동차가 특정 상황이 발생하여 그룹을 이탈하고자 할 때, 그룹에서는 이탈하는 차량에게 통제권을 이양해 줌으로써, 이탈 차량이 독립적인 자율주행을 유지할 수 있도록 해야 한다. 이를 위해서는 그룹 이탈에 대한 차량간의 통신이 이루어지는 것이 중요하다.

이에 본 연구에서는 V2V(Vehicle to Vehicle) 통신[4]을 활용한 그룹 이탈 경고 시스템에 대한 알고리즘을 Wright 언어[5,6]로 명세하여 규격화 된 표기 방법을 제시하고, 이를 State Transition Diagram으로 변환하여 그룹 이탈을 위한 동작이 올바르게 이루어지는 가에 대해 평가한다. 연구의 결과는 군집주행에서의 그룹 이탈을 위한 ADS 개발시 활용할 수 있을 것으로 판단된다.

2. 관련 연구

2.1. V2X 통신

V2X(Vehicle to Everything) 통신이란, 유·무선망 연결을 통해 차량이 주행하면서 도로-자동차-운전자 및 보행자 간 유기적인 상호 작용을 지원하며 교통정보를 교환하거나 공유하기 위한 인프라 기술을 뜻한다.

이 중에서도 V2V 는 이동 중이거나 정차중인 일정 범위 내에 있는 차량들이 무선통신을 통해 각 차량의 주행 정보와 주변 교통 상황 등을 상호 주고받는 통신 기술분야이다. V2V 는 자율주행에서 긴급한 돌발 상황이나, 통신 인프라가 취약한 지역, 또는 통신 량이 과도한 지역에서 유용하게 사용될 수 있는 차량간 통신 방법이다.

2.2. 군집주행

군집주행은 차량 간의 통신을 기반으로 여러 차량이 함께 군집(Platoon)을 형성하여 선두 차량의 교통 상황 정보를 교환하는 기술을 뜻한다. 선두 차량은 추종 차량들과 통신하여 군집의 관리와 운영을 주도하며, 추종 차량은 V2V 통신과 주행환경 정보를 활용하여 차량의 통제권을 선두 차량에게 이양한다. 목적지가 상이하거나 긴급 상황에 의해 군집으로부터 이탈하는 경우, 제어권을 다시 찾아 주행한다. 이러한

군집주행은 자율주행을 위해 요구되는 전체 기능 중에서 일부의 기능만으로도 주행이 가능해진다는 장점과 불필요한 운전을 최소화 시킴으로써, 연료비를 절감할 수 있는 효과를 가져올 수 있다.

2.3. 차선이탈 경고 시스템

차선이탈 경고시스템은 차량 전방에서 획득된 영상정보로부터 차선을 인식하고, 영상정보 및 다른 차량으로부터 수신한 이탈 경보 메시지에 근거하여 차선을 이탈하는지 여부를 판단하여 이탈 알림 메시지를 V2V 통신으로 주변의 다른 차량에 송신하는 시스템[7]이다.

차선이탈 경고시스템은 자차의 위치정보를 획득하는 GPS 모듈과, 자차의 전후방 영상정보를 획득하는 카메라와, V2V 통신부, 그리고 자차 또는 타 차량의 차선 이탈을 판단하고 경보가 발생하도록 하는 차선이탈 제어부, 그리고 경고를 표시하는 디스플레이와 스피커로 이루어져 있다. 차선이탈 경고 시스템의 시스템 구성도는 그림 1 과 같다.

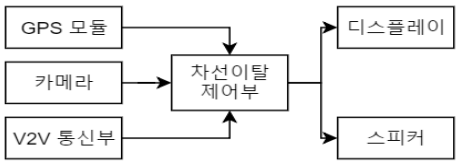


그림 1 차선 이탈 제어 시스템 구성도

2.4. CSP와 Wright

CSP(Communicating Sequential Process)는 시스템을 구성하는 프로세스간의 동작을 명세하고, 프로세스간의 상호작용, 즉 프로토콜을 정의하기 위한 언어[8]이다. 본 연구에서는 CSP 언어의 일종인 Wright 명세언어를 사용하기 이하여 CSP 언어의 의미론적 개념을 활용한다. 본 연구에서 사용하는 CSP의 문법은 다음과 같은 의미를 갖는다.

- Process and Event : Event는 통신 또는 Process간의 상호작용을 의미하고, Process는 event에 참여하는 본체를 의미한다.
- Event §는 성공적인 종료를 나타낸다.
- Prefixing(→) : 특정 event에 참여하여 Process가 변화할 때 사용한다.
- Alternative(□) : Process가 둘 중 하나의 Process로 행동하고, 그 선택이 외부에 의해 결정될 때 사용한다.
- Decision(⊓) : Process가 둘 중 하나의 Process로 행동하고, 그 선택이 외부 환경이 아닌 Process 자신에 결정될 때 사용한다.

Wright는 CSP를 확장한 언어로써, 소프트웨어의 설계를 구성 요소 간의 상호작용으로 명세하기 위한 프로세스 기반 명세 언어이다. Wright 명세를 위해 사용하는 구성 요소[6]는 Component와 Connector, 그리고 Configuror가 있다.

Component는 일련의 Port와 Computation으로 표현할 수 있는데, 각각의 Port는 Component가 그 환경에서 연결된 Component와 상호작용하는 논리적인 동작을 표현한다. 한 Component는 여러 Port를 가짐으로써 시스템의 다른 부분에서 다중 인터페이스를 수행할 수 있게 한다. Computation은 Component가 이루고 있는 환경에서 어떻게 사용되는지에 관계 없이 내부 동작 자체를 설명한다.

Connector는 Role과 Glue 명세로 정의한다. Role은 Component간의 상호 작용에 참여하는 각 참여원의 예상되는 동작을 정의함으로써, 동작 수행을 결정하는 규격으로 사용하며, Glue는 Connector에서 각각의 Role의 활동이 조화하여 수행되는 방식에 대해 설명한다.

Configuror는 각각의 Component에 해당하는 객체를 정의하고, 그 객체에 해당하는 Component의 Port가 어떤 Connector의 Role과 연결되는지를 명세한다. 또한 control 이벤트를 허용함으로써 'new', 'del', 'attach', 'detach'의 동작을 통해 동적으로 객체를 생성 및 삭제하고, Port와 Role을 재구성할 수 있다. 또한, 'where'를 이용하여 각각의 구성요소에서 사용하는 프로세스를 좀 더 한정적이고 상세하게 정의할 수 있다.

3. 군집주행 중 그룹 이탈 제어 알고리즘

군집주행 중에 그룹 내에서 이탈하는 차가 발생할 경우, 이탈 차량의 경보 발생 알고리즘은 그림 2와 같다.

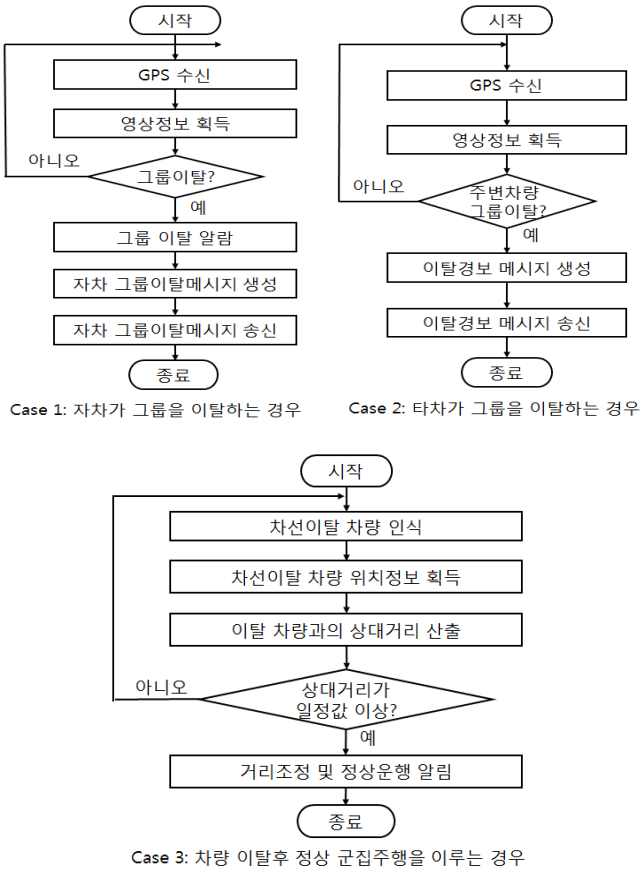


그림 2 군집주행 그룹 이탈 경보 알고리즘

이 알고리즘은 크게 3가지 유형의 행위를 갖는다. 첫 번째(그림 2의 Case 1)는 자차가 그룹을 이탈하여 경보를 발생하는 행위, 다른 차량의 그룹 이탈을 감지하여 알려주는 행위(그림 2의 Case 2), 이탈 메시지를 수신하고, 군집 주행이 계속되는 행위(그림 3의 Case 3)로 구성된다.

우선 Case1에서는 자차 전방에 대한 영상정보를 획득하여 자차가 그룹을 이탈하는지 여부를 판단한다. 이탈한 경우, 자차가 그룹을 이탈했다고 인식한 뒤 자차의 그룹이탈을 알리는 메시지를 생성하여 V2V 통신으로 타 차량에 송신한다. Case2는 영상정보에서 타 차량이 그룹을 이탈했다면 위와 동일한 방식으로 타차의 그룹이탈 경보 메시지를 송신한다. 이렇게 해서 송신된 이탈 메시지를 수신한 차량은, 이탈 차량과의 임의의 거리가 벌어지면(즉, 이탈 차량과 거리가 멀어지면) 자차와 앞차와의 거리조정을 통해 군집주행을 정상적으로 수행하고 정상주행 메시지를 생성한다.

4. 군집주행 이탈 경보 Wright 명세

그림 2의 알고리즘을 통해 각 Case에 대해 메시지 송수신이 어떠한 과정으로 진행되는지 알 수 있다. 하지만 어떠한 상황일 때 해당 알고리즘을 수행하는지, 메시지를 수신받은 개체가 이탈 경보 메시지를 송신할 수 있는지 등의 상호작용에 대한 정보는 확인할 수 없다. 따라서, 이러한 그룹 이탈 과정을 Wright로 명세함으로써 보다 자세한 그룹 이탈 경보 모델을 생성할 수 있다.

그룹 이탈 경보 시스템은 개별적인 차량의 역할을 설명하는 Car Component와 차량간의 경보 알람(Alarm)을 통한 상호작용을 설명하는 Link Connector, 군집 내의 차에 대한 Component의 Port를 Connector의 Role에 연결 및 재구성하는 Configuror로 이루어져 있다. 또한 그룹 이탈이 발생했을 때는 Busy State로, 안전하게 이탈이 완료되면 Idle State로 상태를 구성한다. 그림 3은 자율주행자동차가 수행하는 군집주행 이탈에 대한 컴포넌트의 Wright 명세이다.

그림 3에서 Car의 Port는 자차 또는 타차의 그룹 이탈을 감지하는 Port x와, 그룹 이탈이 발생했을 경우 그룹 이탈 메시지를 수신했을 경우와 그룹 이탈 종료후 정상 메시지를 송신 할 경우에 대해 각각의 역할을 수행하는 Port y로 구성되어 있다.

Port x는 GPS 정보와 영상정보를 수신하여 그룹을 이탈하는 차량을 감지하여 그룹 이탈 알림을 발생시킨다. 또한 Port y는 각각의 Case에 따른 세 가지 Computation의 경우가 존재한다. 첫 번째로, 그룹 이탈 메시지를 수신하고 internalCompute(y_1)을 통해 State를 변경하고, 그룹 이탈 차량을 인식한 뒤 이탈 차량에 대한 정보를 획득하여 그룹 이탈 차량과의 상대 거리를 계산한다. 두 번째로, internalCompute(y_2)을 통해 상대거리가 안정 수치 이상인 경우에 정상 주행 상태를 알리는 메시지를 생성하고 State를 변경한 뒤 정상 주행 메시지를 송신한다. 세 번째는 환기 메시지를 수신하고 internalCompute(y_3)를 통해 State를 변경한다.

Component Car

Port $x = (\overline{ownExitAlarm} \rightarrow x) \square (\overline{othersExitAlarm} \rightarrow x) \square \zeta$

Computation(x)

$= internalCompute(x) \rightarrow (\overline{x.ownExitAlarm} \square \overline{x.othersExitAlarm}) \rightarrow$ Computation(x) $\square \zeta$

where

$internalCompute(x) = GPSReceive \rightarrow ImageInfoReceive \rightarrow CheckExit$

Port $y = (ownExitAlarm \rightarrow y) \square (othersExitAlarm \rightarrow y)$
$\square (ventilate \rightarrow y) \square (\overline{ventilate} \rightarrow y) \square \zeta$

Computation(y)

$= ((y.ownExitAlarm \square y.othersExitAlarm) \rightarrow internalCompute(y_1) \rightarrow$ Computation(y))
$\square (internalCompute(y_2) \rightarrow \overline{y.ventilate} \rightarrow$ Computation(y))
$\square (y.ventilate \rightarrow internalCompute(y_3) \rightarrow$ Computation(y))
$\square \zeta$

where

$internalCompute(y_1) = SwitchToBusyState \rightarrow RecogCar \rightarrow GetInfo \rightarrow ComputeDistance$
$internalCompute(y_2) = MakeVentilationMessage \rightarrow SwitchToIdleState$
$internalCompute(y_3) = SwitchToIdleState$

그림 3 Component Car의 Wright 명세

Connector Link

Role $t_i = (\overline{ownExitAlarm} \rightarrow x) \square (\overline{othersExitAlarm} \rightarrow x) \square \xi \quad , i \in N$

Role $v_j = (ownExitAlarm \rightarrow y) \square (othersExitAlarm \rightarrow y)$
$\square (ventilate \rightarrow y) \square (\overline{ventilate} \rightarrow y) \square \xi \quad , j \in N$

Glue $= \lim_{i \rightarrow N} (\lim_{j \rightarrow N} (t_i.x \rightarrow v_j.y)), i \neq j \rightarrow$ Glue

$\square \lim_{i \rightarrow N} (\lim_{k \rightarrow N-1} (v_i.y \rightarrow v_k.y)) \ , i \neq k \in N \rightarrow$ Glue

$\square \xi$

그림 4 Connector Link 의 Wright 명세

그림 4는 커넥터에 대한 Wright 명세이다. Link의 Role t는 Car의 Port x, Role v는 Port y와 매치된다. 이 때, 같은 Component를 사용하는 여러 인스턴스가 존재하기 때문에 중복된 표기를 줄이기 위해 새로운 표기 방식을 사용하였다. 이 표기 방식은 $\lim_{i \rightarrow k}$ 형식으로 정의하여, i는 1부터 k까지 증가하며 해당 범위 내에 있는 모든 경우에 대해 점선의 상자에 정의된 Event를 수행한다. 이러한 표기법의 사용을 도입함으로써, 군집 내의 N개의 차량 중 한 차량이 그룹 이탈을 시도하였을 때 그 알림이 자차를 제외한 모든 차량에게 메시지를 송신하도록 하게 할 수 있다.

마지막으로, 그림 5는 Exit-Detect Configuror에 대해 명세이다. Car Component에 대한 인스턴스 세 가지와 Link Connector에 대한 인스턴스를 생성한다. 그리고 각각의 Port x와 Port y를 Role t와 Role v에 연결시켜준다. 그리고, 특정 인스턴스가 그룹을 이탈하겠다는 제어 신호를 보내면 해당 Component와 Connector의 연결을 해지하고 인스턴스를 삭제한다. 이처럼 Configuror를 통해 Component와 Connector의 연결을 동적으로 할 수 있고, 인스턴스를 생성 또는 삭제할 수 있다.

$$
\begin{aligned}
&\textit{Configuror}\ \text{Exit-Detect}\\
&\quad new.C_1 : Car\\
&\qquad \rightarrow new.C_2 : Car\\
&\qquad \rightarrow new.C_3 : Car\\
&\qquad \rightarrow new.L : Car\\
&\qquad \rightarrow attach.C_1.x.to.L.t_1\\
&\qquad \rightarrow attach.C_1.y.to.L.v_1\\
&\qquad \rightarrow attach.C_2.x.to.L.t_2\\
&\qquad \rightarrow attach.C_2.y.to.L.v_2\\
&\qquad \rightarrow attach.C_3.x.to.L.t_3\\
&\qquad \rightarrow attach.C_3.y.to.L.v_3\\
&\qquad \rightarrow WaitForChange\\
&\quad \textbf{where}\\
&\quad WaitForChange = \lim_{k \rightarrow N}(C_k.control.GroupExit \rightarrow detach.C_k.x.from.L.t_k\\
&\rightarrow detach.C_k.y.from.L.v_k \rightarrow WaitForChange)
\end{aligned}
$$

그림 5 Configuror Exit-Detect의 Wright 명세

5. 군집주행 이탈 경보 Statement Transition Diagram

4장에서 기술한 Component Car의 Wright 명세에서, Port x의 internalCompute를 이용하여 State를 변경해주었었다. 본 절에서는 군집이탈 경보 알고리즘에 나타난 State 변화에 대한 정보를 기반으로 Statement Transition Diagram을 그림 6과 같이 작성하였다.

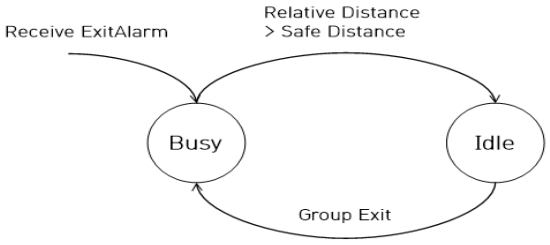


그림 6 그룹 이탈 경보 상태-전이 다이어그램

그룹 이탈 경보를 수신한 경우, Busy 상태로 전이한다. 즉, 군집으로부터 차선의 이탈 현상이 발생하기 때문에 차량간 통신 등의 동작이 발생하는 상태이다. 또한 이탈이 완료된 후에는 정상주행 메시지를 생성 또는 수신하게 되는데, 이때는 이탈 차량과의 거리를 기반으로 Idle 상태로 전이한다. 이는 상대 거리가 일정 거리 이상의 안전한 거리일 경우에 해당한다. Idle 상태에서 다시 그룹 이탈이 발생할 경우에는 Busy 상태로 전이한다.

6. 결론

본 연구에서는 V2V 통신을 활용한 군집주행 그룹 이탈 경보 시스템에 대한 이탈 알고리즘을 Wright 언어로 명세하고, 이를 State Transition Diagram으로 나타내었다. 자율주행자동차가 군집주행 중일 때 추종차량이 그룹을 이탈하였을 때의 각 차량에 대한 동작 알고리즘을 Wright로 명세함으로써 주행중의 그룹 이탈 상황에 대한 각 차량간에 상호작용하는 과정을 확인할 수 있다.

또한 Wright 명세를 바탕으로 상태-전이 다이어그램을 생성이 가능함을 확인하여 해당 알고리즘에 대한 테스트를 진행할 수 있을 것이라 기대한다.

향후의 연구로써 자율주행자동차의 제어 분야에 대한 다양한 기능을 Wright 언어로 명세해 보고, ADS를 구성하는 각 컴포넌트간의 상호 작용간 발생할 수 있는 충돌 현상이나 문제점들을 발견하는 연구로 확장시킬 예정이다.

Acknowledgement

본 논문은 교육부가 지원하고 충북대학교 컨소시엄에서 주관하는 2018학년도 지역선도대학육성사업의 지원을 받아 수행한 것임.

참고 문헌

[1] 이승민, "자율주행자동차 최근 동향 및 시사점", ITFIND 주간기술동향, 2018.4.18

[2] 오원석, "구글 자율주행차, 첫 사고 기록..'책임 인정'", 블로터, http://www.bloter.net/archives/251067, at 2018.12.07

[3] 한겨레, "현대차그룹-오로라 “2021 년 자율주행 레벨 4 기술 상용화", 홍대선, 2018.01.04

[4] 임태호, “자율주행과 V2X 통신 기술 동향”, ITFIND 주간기술동향, 2017.11.1

[5] Robert Allen, A Formal Basis for Architectural Connection, ACM TOSEM, 6(3), 1997, pp.213-249

[6] Robert Allen, Specifying Dynamism in Software Architectures, Proceedings of the Workshop on Foundations of Component-Based Software Engineering 1997.09

[7] 노동규, “차선 이탈 경보 방법, 장치 및 시스템”, 대한민국특허청, 2012.12.12

[8] Stephen D. Brookes, C. A. R. Hoare, A. W. Roscoe, “A Theory of Communicating Sequential Processes, Journal of ACM, 1984.

빅데이터 분석용 DBMS를 위한 테스트 자동화 프레임워크 설계

임우섭○, 송옥수, 김현정
한국정보통신기술협회(TTA)
woosup.lim@tta.or.kr, donut92@tta.or.kr, hchkim@tta.or.kr

Test Automation Framework for OLAP DBMS

Woo Sup Lim○, Oksu Song, Hyunchong Kim
Telecommunications Technology Association

요 약

빅데이터 분석용 DBMS는 대량의 데이터를 처리, 저장 및 분석하는 시스템으로, 단순 결함도 치명적인 장애로 이어질 수 있는 높은 신뢰도가 요구되는 시스템이다. 고신뢰성의 DBMS를 개발하기 위해서는 다양한 테스트 방법론을 적용하고 방대한 테스트케이스를 구축하여 체계적인 테스트를 수행해야 한다. DBMS 제품 특성상 기능이 추가될 때마다 전체 기능에 대한 테스트를 수행해야 하고, 운영체제, 설정파일 등을 변경하는 경우에도 DBMS의 전체 기능에 대한 회귀 테스트는 필수적이다. 동시성 시험 등 수작업으로 테스트가 불가능한 영역이 다수 있기 때문에 시험 자동화에 대한 필요성은 더욱 높아진다.

본 논문에서는 체계적인 테스트 수행을 위해 DBMS 개발기업이 필요로 하는 테스트 자동화 프레임워크를 제안하고자 한다. SW개발기업의 실질적인 활용을 위해 기업의 요구사항을 분석하여 프레임워크를 설계하였으며 본 논문에서는 설계 결과인 구성 요소 및 요소별 설계 내용을 소개하고자 한다.

1. 서 론

한국데이터진흥원 조사 결과[1]에 따르면 빅데이터 솔루션 기업의 국내 기술 수준은 2016년 기준 선진국 대비 약 65.7점, 기술수준 격차는 약 3.1년, 선진 기술 도달 시간은 약 3.4년으로 큰 차이를 보이고 있다. 외산 제품 대비 경쟁력 저하의 주요 원인으로는 국내 제품의 낮은 안정성 및 신뢰성 등으로 분석하고 있다.

빅데이터 분석용 DBMS는 대량의 데이터를 처리, 저장 및 분석하는 시스템으로, 단순 결함도 치명적인 장애로 이어질 수 있기 때문에 신뢰성을 가장 중요하게 고려하고 있다. 고신뢰도를 요구하는 DBMS의 경우, 소스코드를 일부만 수정하여도 전체 시스템에 대한 전수 시험을 수행하고, 매뉴얼 테스트가 불가능한 기술적인 영역이 존재하기 때문에 테스팅 도구 활용은 필수적이다.

소프트웨어를 비롯하여 다양한 산업 분야에서 반복적인 작업을 줄이고 업무의 효과를 높이기 위해 자동화를 추진하고 있다. 다양한 테스팅 영역에서 테스팅이 자동화되고 있지만, 쉽게 변경이 가능한 소프트웨어 특성으로 인해 성능시험도구, GUI 테스팅 등 제한된 영역을 대상으로 테스팅 자동화가 이루어지고 있다.

개발된 제품을 대상으로 일차적인 테스팅이 완료되면 그 이후부터는 반복적인 테스팅 작업이 시작된다. 결함 패치, 기능 업그레이드 등 일부 기능이 변경된 경우에도 제품의 품질을 확보하기 위해서는 전체 기능에 대한 시험이 필수적이다. SW개발기업에서는 이러한 반복적인 테스트를 효과적이고 생산적으로 수행하기 위해 테스팅 자동화 도구를 도입한다. 테스팅 자동화는 반복된 테스팅 작업에 대한 투입 대비 효과를 높일 수 있고 수작업으로 테스팅이 불가능한 기술 영역도 테스트가 가능하도록 도와줄 수 있다. 특히, DBMS, 운영체제 등과 같이 높은 신뢰성을 요구하는 시스템 소프트웨어 분야의 경우, 수작업으로 테스팅이 불가능한 영역이 존재하기 때문에 도구의 필요성은 더욱 높아진다.

DBMS의 복잡성으로 인해 DBMS에서 결함 발생 시, 결함이 발생한 상황을 수작업으로 재현하는 것은 현실적으로 매우 어려운 일이다. 그렇기 때문에 테스트 과정을 실시간으로 로그로 기록하는 등 일련의 작업을 자동화해야한다. DBMS가 다수 사용자의 동시 요청에 대한 제어를 제대로 처리하지 못한다면, 데이터의 무결성이 깨질 수 있는 치명 결함이 발생할 수 있기 때문에 이를 테스트할 수 있는 시험 도구는 테스트의 핵심이 된다. 이러한 배경과 함께 본 논문에서는 DBMS 개발기업의 효과적인 품질 검증을 지원하기 위해 테스팅 자동화 프레임워크 연구개발을 제안하였다.

단순 결함도 치명적인 장애로 이어질 수 있는 시스템 특성상 제품 신뢰성에 대한 기대수준이 매우 높은 상황이다. 고신뢰성의 DBMS를 개발하기 위해서는 다양한 테스트 방법론의 적용은 물론이고 방대한 테스트케이스를 구축하여 완전한 테스트를 수행하여야 한다. DBMS 제품에 기능이 추가될 때마다 전체 기능에 대한 테스트를 수행해야 하고, 운영체제, 설정파일 등을 변경하는 경우에도 DBMS의 전체 기능에 대한 테스트는 필수적이다. 다수 사용자 동시 요청과 같은 테스트케이스는 수작업으로 테스트 수행이 불가능하기 때문에 테스팅 자동화 도구는 DBMS 검증에서는 없어서는 안되는 필수품이라는 사실은 두말할 것도 없다.

본 논문에서는 DBMS 개발기업이 필요로 하는 테스트 자동화 프레임워크를 제안하고자 한다. SW개발기업에서 실질적으로 활용할 수 있도록 기업의 요구사항을 분석하여 프레임워크를 설계하였고, 프레임워크 설계 결과인 구성 요소 및 각 요소별 설계 내용을 소개하고자 한다. 테스트 자동화 프레임워크는 4가지 요소로 구성되고, 각각은 다음과 같다.

① 테스트 슈트 레파지토리
② 자동화 관리 엔진
③ 자동화 실행 엔진
④ 결과 확인 및 분석을 위한 시각화 도구

본 논문의 구성은 다음과 같다.

2장 배경
3장 자동화 프레임워크 필요성 및 설계 고려사항
4장 빅데이터 분석용 DBMS 테스트 자동화 프레임워크
5장 결론 및 향후 계획

2. 배 경

2.1 기존 DBMS 테스트 자동화 도구

고신뢰성이 요구되는 DBMS의 경우 상용 제품뿐만 아니라 오픈소스에서도 테스트 자동화 프레임워크를 활용하고 있다. 대표적인 테스트 자동화 프레임워크는 다음과 같다.

1) MySQL Test Framework

MySQL 서버와 클라이언트 프로그램을 테스트하기 위해 제공되는 MySQL 테스트 프레임워크로[2], 테스트를 위한 별도의 언어를 제공하는 것이 특징이다. 테스트 언어에 따라 작성된 테스트 슈트가 MySQL과 함께 배포되고 있어 MySQL을 대상으로 테스트 수행이 가능하다. 테스트 프레임워크는 다수의 프로그램으로 구성되고 SQL문 테스트, 클라이언트 API 테스트, 부하 테스트, 단위 테스트 등이 가능하다. 하지만 MySQL에 특화되어 있기 때문에 다른 DBMS를 대상으로 테스트를 수행하는 것은 불가능하며 별도의 언어를 가지고 있어 기능 추가 시 개발 비용이 큰 편이다.

2) Testgres

PostgreSQL을 테스트하기 위한 테스트 프레임워크로[3], 테스트케이스가 파이썬 파일로 작성되는 것이 특징이다. 작성된 테스트케이스는 일반 파이썬 파일과 동일하게 파이썬 인터프리터를 통해 실행된다. 파이썬을 이용하기 때문에 파이썬에서 제공하는 제어문과 풍부한 라이브러리를 가져다가 사용할 수 있다는 장점을 갖는다. 하지만, MySQL 테스트 프레임워크와 달리 실행 결과를 검증하는 메커니즘을 제공하지 않아 사용자가 테스트케이스 내에서 검증을 진행해야 하며, 테스트 슈트의 개념이 없어 테스트케이스를 하나씩 실행해야 하는 단점을 갖는다. MySQL과 마찬가지로 다른 DBMS에 대한 테스트는 불가능하다.

3) Oracle Real Application Testing

상용 소프트웨어인 오라클은 DBMS를 테스트할 수 있는 솔루션을 판매하고 있다. 이중 Real Application Testing(RAT) 솔루션[4]은 DB의 운영환경 변경, 버전 업그레이드, 파라미터 변경 등에 따른 전체 기능 테스트 및 부하 테스트가 가능한 솔루션이다. DB Replay 기능을 통해 운영 환경에서의 워크로드를 저장하여 동시성 및 트랜잭션을 테스트 시스템에 재현할 수 있으며, 성능 분석기를 통해 SQL 실행계획 및 통계정보를 검증할 수 있다.

3. 자동화 프레임워크 필요성 및 설계 고려사항

3.1 DBMS 자동화 프레임워크 필요성

고신뢰성이 요구되는 DBMS 특성상 상용 SW 뿐만 아니라 오픈소스에서 테스트 자동화 프레임워크를 함께 제공하고 있다. 테스트 자동화 프레임워크가 활용되고 있는 이유는 다음과 같다.

① [5]에 따르면 데이터베이스 테스트 자동화는 약 2,400배의 능률을 향상시키는 효과가 있다. 자동화를 통해 테스터는 반복적인 회귀 테스트에서 벗어나 새로운 테스트 개발과 도구 개발에 집중할 수 있으며 결과적으로 제품의 품질을 높일 수 있다.

② 환경 설정이나 결함 패치 시에 테스트를 반복적으로 수행해야 하는 상황으로 자동화 도구 없이 수작업으로 하는 것은 비용적으로 효과적이지 않다. 단순 결함 패치도 시스템 장애로 이어질 수 있기 때문에 패치에 대한 영향 분석을 실시해야 하고 이를 위해서는 변경 시마다 전수 테스트를 수행해야 한다.

③ DDL, DML, DCL 등 조합에 따라 수많은 SQL 문이 존재하며 이를 수동 테스트를 통해서 실행 및 검증하는 것은 불가능하다.

④ DBMS에서 제공하는 동시성, 이중화 등 기능의 경우 매뉴얼 테스트가 불가능하다. 예를 들어 동시성 테스트를 위해서는 다수 사용자의 동시 요청을 시뮬레이트해야 하지만 도구 없이는 불가능하다.

⑤ DBMS는 다수의 모듈이 유기적으로 동작하는 복잡한 시스템으로 오류 발생 시 원인 파악이 매우 어렵다. 원인분석을 위해서는 시험 과정을 자동으로 기록하는 자동화 도구가 필수적이다.

⑥ DBMS는 시스템의 자원을 효율적으로 활용해야 하는 시스템 소프트웨어로, 각 기능의 자원(CPU, 메모리, 디스크) 사용률을 분석하기 위해 자원 모니터링 도구와의 연동이 필요하다.

3.2 DBMS 자동화 프레임워크 요구사항

본 논문에서는 중소기업에서 개발한 빅데이터 분석용 DBMS의 품질 고신뢰성 확보를 위해 테스트 자동화 프레임워크 연구 개발을 제안하고자 한다. 해당 기업에게 수집한 요구사항은 아래와 같다.

① 자동화 테스트 실행 시, 선행 테스트 케이스로 인해 DBMS가 비정상 중지되더라도 중단없이 이후 테스트가 수행되어야 한다.

② 동시성 테스트를 위해 다중 세션을 지원하고 트랜잭션 제어(시작, 커밋, 롤백 등)가 가능해야 한다.

③ 다양한 DBMS 운영 환경(단독운영, 이중화, DBLink 등) 구성을 자동화할 수 있어야 한다.

④ 운영 간 발생하는 성능 문제(메모리 누수, 응답 시간 지연 등)를 탐지할 수 있어야 한다.

3.3 프레임워크 설계 고려사항

본 논문은 DBMS 개발 기업의 요구사항을 기반으로 시작되었지만, 다양한 DBMS에 대한 테스트 자동화에 적용할 수 있도록 프레임워크를 설계하였다. 요구사항을 기반으로 프레임워크 설계 시 고려사항은 다음과 같으며 각 고려사항은 [표 1]과 같이 설계 항목에 반영하였다.

① One-Stop 테스트: 한번의 실행으로 DBMS의 전체를 테스트할 수 있도록 자동화. 테스트 과정에서 에러가 발생하여도 중단없이 테스트가 진행되도록 에러 처리

② 다중 세션: 동시에 여러 세션을 생성하여 SQL문을 실행하고 트랜잭션 제어가 가능하도록 기능 지원

③ 운영 환경 구성: 다수 노드에서 DBMS 서버 실행, 이중화 설정, DB Link 설정 등이 가능하도록 관리 기능 제공

④ 성능 모니터링: CPU 사용률, 메모리 사용량, SQL문 수행 시간을 모니터링하고 사용자가 명시한 기준 범위를 벗어날 경우 해당 테스트 케이스를 실패 처리하는 기능 제공

⑤ 테스트 오라클: 테스트 케이스별로 성공 실패를 비교할 수 있는 메커니즘을 제공. 기대 출력과 실제 출력을 비교하여 성공 실패를 결정하고, 공백 문자열, 날짜 차이 등으로 발생할 수 있는 오류에 대한 전처리

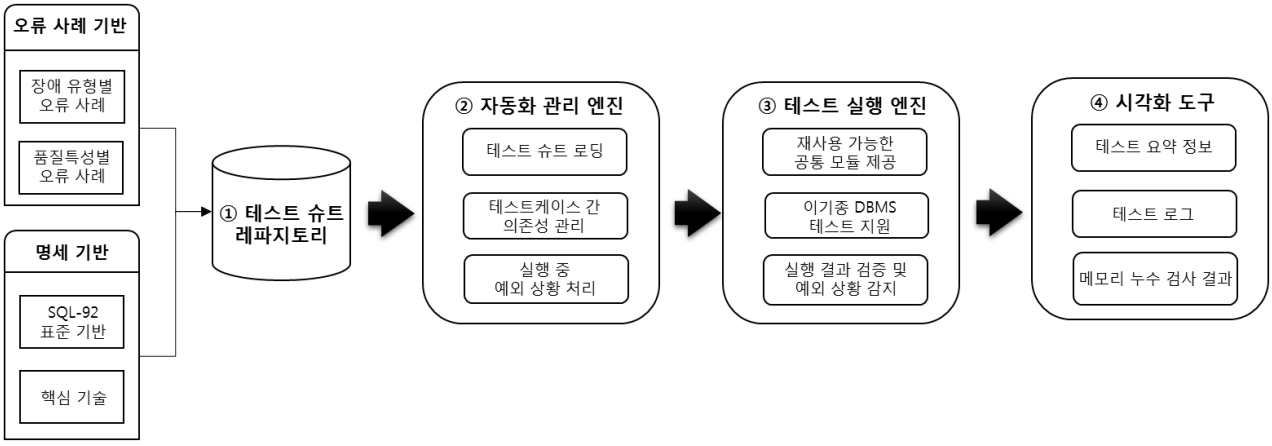
⑥ 유지보수 용이성: DBMS 기능이 추가됨에 따라 테스트케이스가 지속적으로 확장될 수 있기 때문에 테스트케이스 추가, 수정 및 삭제가 용이하도록 설계

⑦ 형상 관리: DBMS의 형상 변경에 따라 테스트케이스의 형상도 함께 변경되어야 하므로, 자동화 프레임워크와 형상 관리 도구와의 연동 고려

⑧ 의존성 설정: 테스트케이스의 실행 의존성을 고려하여 테스트가 실행될 수 있도록 설계

[표 1] 설계 고려사항별 설계 결과

요구사항 및 설계 고려사항	설계 결과
One-Stop 테스트	자동화 관리 엔진에서 테스트 실행 중 DBMS 비정상 종료와 같은 예외 상황 발생 시, 자동으로 DBMS 서버를 초기화하도록 설계
다중 세션	다중 세션 생성 기능을 제공하는 동시성 지원 패키지를 테스트 실행 엔진에 설계
운영 환경 구성	이중화, DB Link 등의 운영 환경 구성을 자동화 할 수 있는 운영환경 제어 패키지를 테스트 실행 엔진에 설계
성능 모니터링	자원 모니터링 패키지를 테스트 실행 엔진에 설계
테스트 오라클	자동화 관리 엔진에서 실행 결과 검증 모듈을 통해 테스트 오라클 설계
유지보수 용이성	테스트케이스 추가, 수정 및 삭제가 용이하도록 텍스트 파일 단위로 테스트 슈트 구조 설계
형상 관리	형상관리도구와 연동을 통해 DBMS 제품 형상별 테스트케이스 관리 및 추적 지원
의존성 설정	테스트 슈트 옵션을 통해 의존성을 설정할 수 있도록 설계



[그림 1] 빅데이터 분석용 DBMS 테스트 자동화 프레임워크

4. 빅데이터 분석용 DBMS 테스트 자동화 프레임워크

본 논문에서는 [그림 1]과 같은 빅데이터 분석용 DBMS 테스트 자동화 프레임워크를 제안하고자 한다. 자동화 프레임워크는 크게 1) 테스트 슈트 레파지토리, 2) 자동화 관리 엔진, 3)테스트 실행 엔진, 4)시각화 도구로 구성되고 각각에 대한 설명은 다음과 같다.

4.1 테스트자동화 프레임워크 구성

1) 테스트 슈트 레파지토리

테스트 슈트를 저장하고 관리하기 위한 저장소로, 하나의 테스트 슈트는 다수의 테스트케이스로 구성되어 있다. 테스트케이스는 파이썬의 모듈 단위로 작성하고 테스트 슈트는 다수의 파이썬 모듈 파일을 묶어서 실행하도록 레파지토리를 설계하였다. 테스트 슈트 레파지토리에는 기대 출력과 실제 출력을 비교할 수 있도록 기대 출력을 정의한 파일이 포함되어 있다.

2) 자동화 관리 엔진

테스트케이스 실행을 자동화하기 위한 관리 모듈로, 테스트케이스 로딩, 실행 순서 결정, 테스트 실행 모듈에 실행 요청, 실행 결과 기록 등을 수행하는 엔진이다. 또한, 테스트케이스가 중단없이 실행될 수 있도록 예외 처리 기능을 제공한다.

3) 테스트 실행 엔진

테스트케이스를 실행하고 실행 과정을 로그로 기록하는 모듈로, 실행하는 과정에서 비정상 종료와 같은 예외 상황 발생 시 이를 감지한다. 또한, 각 테스트케이스 실행 과정에서 기록된 실제 출력과 레파지토리의 기대 출력을 비교하여 성공/실패를 결정한다.

4) 시각화 도구

테스트 실행 결과(테스트케이스 실행 건수, 실패 및 성공 건수 등)와 실패 테스트케이스에 대한 상세 로그 조회, 메모리 누수 여부 검사 결과 조회 등을 시각적으로 제공한다.

4.2 테스트 자동화 프레임워크 설계

특정 기업의 빅데이터 분석용 DBMS 뿐만 아니라 다른 기업의 제품에도 활용될 수 있도록 확장성과 호환성 있는 테스트 자동화 프레임워크를 설계하였다. 또한, 테스트케이스를 지속적으로 추가 및 관리할 수 있도록 유지보수를 고려하여 시스템을 설계하였다. 프레임워크의 각 구성 요소별 설계 내용은 다음과 같다.

4.2.1 테스트 슈트 레파지토리

1) 테스트 슈트 레파지토리 구조

일반적으로 적용되는 디자인 패턴에 따라 테스트케이스를 디렉토리 기반으로 구조화하였고, 논리적으로 성격이 유사한 테스트케이스를 하나의 테스트 슈트로 그룹핑하였다. DBMS에 대한 체계적인 테스트를 위해 테스트 슈트는 아래와 같이 분류하였고, 각각의 테스트케이스는 파이썬으로 개발하였다.

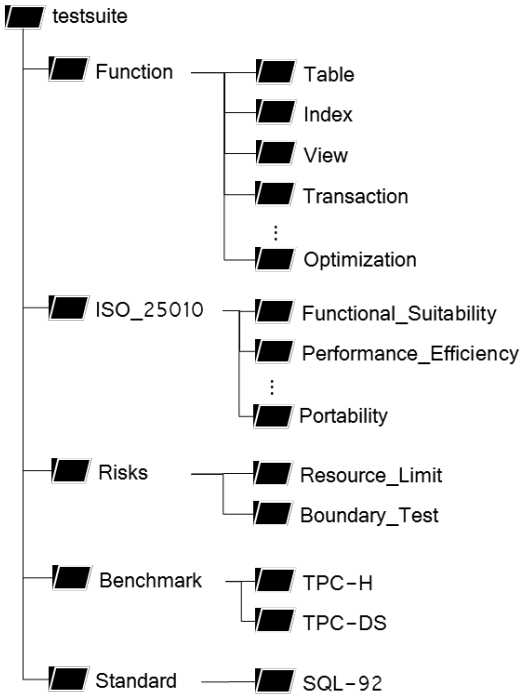
① 제품 기능: 테이블, 인덱스, 뷰, 트랜잭션, 데이터 타입, 유저 및 권한, 프로시져, 함수, 최적화 등

② ISO/IEC 25010 기반 품질 특성[6]: 기능적합성, 성능효율성, 호환성, 사용성, 신뢰성 등

③ SQL-92 표준: 데이터 정의어, 데이터 조작어, 트랜잭션 관리, 커넥션 관리, 세션 관리, 동적 SQL 등

④ 장애유형: 자원 한계 상황, 경계값, 네트워크 절체, 비정상 종료, 비정상 중지 등

⑤ 벤치마크테스트: TPC-C[7], TPC-H[8], TPC-DS[9] 등



[그림 3] 테스트 슈트 레파지토리 구조 예시

[그림 3]와 같이 테스트 슈트는 디렉토리와 매핑되도록 구성하였고, 테스트 슈트명을 디렉토리명으로 사용하였다. 테스트케이스 추가, 삭제, 수정 등의 관리가 용이하도록 텍스트 파일 단위로 테스트 슈트를 작성하였다. DBMS에 새로운 기능이 추가되는 경우 이에 대한 테스트케이스를 추가해야 한다. 테스트 자동화 프레임워크에서는 디렉토리 및 파일을 추가하는 것만으로 테스트케이스 추가가 가능하도록 유지보수 용이성을 고려하였다.

2) 테스트 슈트명 기반 테스트 실행

테스트를 전체 혹은 부분을 수행할 수 있도록 최상위 슈트가 다수의 슈트를 포함하고, 하나의 슈트는 다수의 테스트케이스를 포함하도록 설계하였다. 슈트를 선택하여 부분 테스트가 가능하도록 하였고, 최상위 슈트를 선택하면 전체 기능에 대한 테스트가 자동으로 실행된다.

3) 테스트케이스 형상 관리

테스트케이스가 지속적으로 변경될 수 있기 때문에 소스코드 형상관리처럼 테스트케이스에 대한 형상관리도 중요하다. 이러한 이유로 프레임워크 설계 시 형상관리 도구와 연동하여 테스트케이스를 관리하고 테스트 변경 이력을 프레임워크의 시각화 도구에서 조회할 수 있도록 설계하였다.

DBMS의 경우에는 고객사별, 버전별, 운영체제별로 다양한 형상이 존재하기 때문에 이에 대한 테스트케이스도 제품의 형상에 맞게 관리되어야 한다. 따라서 SW개발기업에서

```
###############################################
#                  기능 테스트 케이스                  #
###############################################

table.suite
index.suite
view.suite
transaction.suite
optimization.suite
replication.suite  --init=create_repl.py  --fini=destroy_repl.py
dblink.suite  --init=create_dblink.py  --fini=destroy_dblink.py
utility.suite
interface.suite
```

[그림 2] 테스트케이스 의존성 설정 예

사용하는 형상관리도구(Subversion 등)와 연동함으로써 DBMS 제품 형상별 테스트케이스 관리 및 추적이 용이하도록 하였다.

4) 테스트 오라클

테스트 오라클이란 기대 출력과 실제 출력을 비교해주는 메커니즘으로, 방대한 테스트를 수행하는 DBMS 테스트를 위해서는 반드시 구현되어야 할 요소이다. 이를 위해 각 테스트케이스별 기대 출력을 인벤토리로 구축하였고 테스트 실행 결과와 기대 출력을 자동으로 비교할 수 있도록 메커니즘을 설계하였다.

4.2.2 자동화 관리 엔진

1) 테스트 슈트 로딩

테스트 슈트 단위로 테스트가 실행되기 때문에 자동화 관리 엔진이 테스트 슈트를 로딩하고, 해당 슈트에 포함된 테스트케이스를 목록화 하도록 설계하였다.

2) 테스트케이스 간 의존성 관리

다수의 테스트케이스는 순차적으로 실행된다. 두 개 이상의 테스트케이스가 상호 의존 관계에 있을 경우, 선행 테스트케이스를 우선 실행하고 후행 테스트케이스를 나중에 실행하도록 자동화 관리 엔진을 설계하였다. 일부 테스트케이스는 테스트케이스의 사전 조건으로도 활용될 수 있기 때문에 이와 같은 테스트케이스는 의존성 관계에서 선행 테스트케이스로 정의하였다. 예를 들어, 사전조건으로 활용될 수 있는 테스트케이스는 [그림 2]과 같이 이중화 설정, DB Link 설정 등이 될 수 있다.

3) 실행 중 예외 상황 처리

테스트를 수행하다보면 DBMS의 결함으로 DBMS가 비정상 종료 혹은 중지될 수 있다. DBMS가 비정상 종료 및 중지될 경우, 자동화 프레임워크의 테스트도 함께 중단되기 때문에 DBMS의 재구동이 필요하다. DBMS의 비정상 종료 및 중지

상황 발생 시, [표 2]과 같이 테스트 정보를 백업하고 데이터베이스를 초기화한 후 서버를 재구동하도록 자동화 관리 모듈을 설계하였다. DBMS가 재구동이 완료되면 중단된 시점부터 테스트를 다시 수행하도록 하였다.

[표 2] 백업 파일 대상

백업 대상	설 명
Core 덤프 파일	메모리에 저장된 프로세스 상태 정보를 백업하여 비정상 종료 시 메모리의 저장 데이터를 분석하기 위해 활용
Stack Trace	프로세스 실행과정을 추적할 수 있도록 콜스택 정보를 백업하여 비정상 중지를 발생시킨 함수, 쿼리를 분석하기 위해 활용
DBMS 로그 파일	DBMS 운영 로그, 에러 로그 등을 백업하여 예외 상황 발생 시 DBMS의 실행 기능을 파악하기 위해 활용

4.2.3 테스트 실행 엔진

1) 재사용이 가능한 공통 모듈 제공

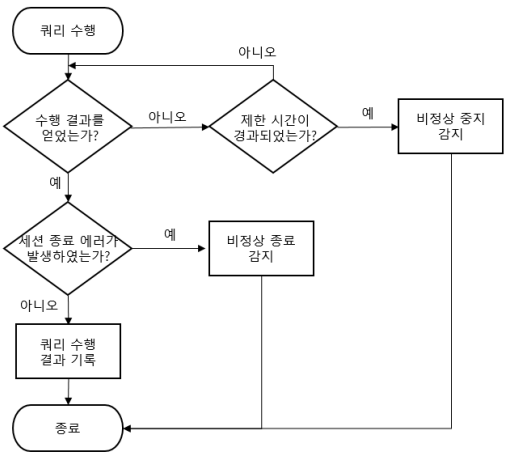
DBMS 테스트 수행 시 테이블 추가, 데이터 삽입, 서버 재구동 등의 작업이 반복적으로 발생한다. 반복적으로 실행되는 작업을 재사용할 수 있도록 공통 작업을 모듈화하였다. 테스트 실행 엔진을 구성하는 공통 모듈은 다음과 같다.

① DBMS 서버 제어 패키지: 데이터베이스 생성, 구동, 중지, 재시작 등 데이터베이스 및 서버에 대한 제어 기능을 제공하는 패키지
② 동시성 지원 패키지: 다수 사용자의 동시 요청을 시뮬레이트하기 위한 패키지
③ 운영환경 제어 패키지: 이중화, DB Link 등 DBMS 운영환경 설정을 위한 패키지
④ 자원 모니터링 패키지: 자원(CPU, 메모리, 디스크) 사용량을 모니터링하는 패키지
⑤ 벤치마크 패키지: TPC-H, TPC-DS 표준에서 제시하는 쿼리 실행용 패키지

2) 이기종 DBMS 테스트 지원

특정 DBMS 뿐만 아니라 다양한 DBMS에 대한 품질 검증을 지원하기 위해 실행 엔진 설계 시, 아래 사항을 고려하였다.

① 다양한 DBMS로 접속이 가능하도록 JDBC 활용
② DBMS별 상이한 쿼리문 지원을 위해 주요 DBMS에 대한 쿼리문 생성 함수 제공



[그림 4] 예외 상황 감지 메커니즘

③ DBMS별 상이한 명령어 지원을 위해 주요 DBMS에 대한 명령어 래퍼 지원

3) 실행 결과 검증 및 예외 상황 감지

테스트케이스가 실행되면 레파지토리의 기대 출력과 테스트 실행 결과를 비교하여 각 테스트별 성공/실패를 결정한다. 또한, 실패된 테스트케이스에 대한 사유를 확인할 수 있도록 로그 조회 기능을 제공한다. 기대 출력과 실제 출력을 텍스트로 비교할 경우 날짜 포맷 차이, 공백 등으로 인해 테스트케이스 실행 결과가 '실패'로 잘못 판단될 가능성이 높기 때문에 정규표현식을 사용하여 이를 처리할 수 있도록 하였다.

테스트케이스 실행 중에 DBMS가 비정상 종료 또는 중지될 수 있기 때문에 [그림 4]와 같이 예외 상황 감지 메커니즘을 설계하였다. 테스트케이스 실행 후 일정시간 이내에 DBMS에 응답이 없을 경우 관리 엔진을 호출하여 DBMS를 재구동 시킨다.

4) 메모리 누수 검출을 위한 Valgrind와 연동

DBMS는 장시간 중단없이 구동되어야 하는 시스템 소프트웨어로 작은 메모리 누수도 시스템 장애로 이어질 수 있는 위험이 있다. 그렇기 때문에 DBMS 실행 과정에서 발생되는 메모리 누수를 사전에 검출할 경우 신뢰성 향상에 큰 도움이 된다. 메모리 누수 검출을 위해 많이 사용되는 Valgrind 프로그램[10]을 연동하였고 메모리 누수 상황을 쉽게 파악할 수 있도록 메모리 누수 검출 결과를 테스트케이스와 맵핑하였다.

4.2.4 시각화 도구

테스트 실행 결과를 시각적으로 분석할 수 있도록 시각화 도구를 제공하였고 다음의 정보를 시각적으로 제공한다.

① 테스트 요약 정보: 테스트 수행 정보(수행 환경, 제품 형상, 테스트케이스 형상 등)와 실행 결과에 대한 통계 정보(성공/실패 수, 실패된 테스트케이스 리스트)
② 원인 분석을 위한 테스트 로그: DBMS 로그 및 실패 테스트케이스에 대한 상세 로그
③ 원인 분석 결과 및 공유: 실패 테스트케이스에 대한 원인 분석 결과
④ 메모리 누수 검사 결과: 메모리 누수 상세 내용 및 관련 테스트케이스

5. 결론 및 향후 계획

새로운 비즈니스 기회와 부가가치를 창출하기 위해 빅데이터 활용 중요성이 높아지고 있다. 이에 따라 빅데이터 분석용 DBMS에 대한 시장의 니즈는 매년 증가하고 있고, 다양한 빅데이터 분석용 DBMS 제품들이 출시되고 있다. 이러한 변화에 발 맞추어 국내 기업들도 빅데이터 분석을 위한 DBMS를 출시하고 있으나, DBMS의 복잡한 구조, 다양한 운영 환경, 방대한 기능 등으로 인해 신뢰성 높은 DBMS 개발에 어려움을 겪고 있다.

DBMS와 같이 복잡하고 고신뢰성이 요구되는 시스템 소프트웨어의 경우 테스팅 인력만으로는 체계적인 테스트 수행에 한계가 있다. 이러한 이유로 빅데이터 분석용 DBMS 개발기업에서 TTA에 테스트 자동화 프레임워크 개발을 제안하였다. 본 논문에서 제안한 자동화 프레임워크는 실 사용자의 요구사항을 분석하여 설계하였고, 다양한 DBMS를 대상으로도 활용될 수 있도록 변경성과 확장성을 고려하여 설계하였다. 또한, DBMS에 새로운 기능이 추가될 때마다 테스트케이스도 함께 추가되어야 하므로 테스트케이스 추가, 수정 및 삭제가 용이하도록 프레임워크를 설계하였다. 그리고 테스트 결과를 시각적으로 제공함으로써 DBMS의 결함 발생현황을 직관적으로 확인할 수 있도록 하였고 테스트 케이스의 오류 발생 시 원인 분석이 가능하도록 상세한 로그를 제공하였다.

현재, 본 논문에서 제안한 소프트웨어 아키텍처를 기반으로 테스트 자동화 프레임워크를 개발하고 있다. 프레임워크 개발이 완료되면 DBMS개발기업에 이를 설치하여 빅데이터 분석용 DBMS를 대상으로 시범적용을 수행할 예정이다.

* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 20180011930012002. 빅데이터 솔루션의 고신뢰성 품질 확보를 위한 품질 검증 자동화 프레임워크 개발)

참 고 문 헌

[1] **2017 데이터산업 백서**. 한국데이터진흥원
[2] The MySQL Test Framework. Available: dev.mysql.com/doc/dev/mysql-server/8.0.1/PAGE_MYSQL_TEST_RUN.html
[3] Testgres. Available: postgrespro.github.io/testgres
[4] Oracle. “Real Application Testing”. Available: www.oracle.com/technetwork/database/options/real-application-testing-option-2355173.html
[5] 도로시 그레이엄, 마크 퓨스터. **소프트웨어 테스트 자동화**. 여용구, 김윤명, 황영석, 성찬혁 옮김. 에이콘, 2013.
[6] ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models
[7] TPC-C. Available: www.tpc.org/tpcc
[8] TPC-H. Available: www.tpc.org/tpch
[9] TPC-DS. Available: www.tpc.org/tpcds
[10] Valgrind Home. Available: www.valgrind.org

Automatic Usage Profiling을 통한 초기 앱 실행 속도 개선 방법

채향석O, 백종문
LG전자, 한국과학기술원
{jen211, jbaik}@kaist.ac.kr

Improving application startup time by automatic profiling

Hyangseok ChaeO, Jongmoon Baik
LG Electronics, Korea Advanced Institute of Science and Technology

요 약

Google은 2009년 Bytecod로 구성된 Dex(Dalvik Executable)를 Dalvik Runtime을 통해 Interpreter로 실행하는 Android 초기 버전을 공개하였다. 이후 성능을 높이기 위해 JIT(Just-in-time)컴파일이 적용되었으며 2015년 Lollipop(Android 5.0)이후로 Dex를 Native Code로 AOT(Ahead-of-time)컴파일하는 ART Runtime이 현재까지 사용되고 있다. ART Runtime이 공개된 초기에는 앱 설치 시점에 모두 AOT컴파일을 수행하도록 동작하였으나 앱 설치 시점에 배터리 소모, 발열로 인한 느림과 사용 공간의 증가로 인해 현재 최신 Android는 AOT컴파일을 앱 설치/업데이트 시점에 모두 수행하지 않고 사용자의 실제 사용 기록인 Profile에 기반하여 나중에 동작한다. 따라서 최신 Android는 앱 설치 이후에 실제 사용 기록에 따른 AOT컴파일이 완료되기 전까지는 앱 실행 속도가 느린 문제가 있다. 이 연구에서는 이 문제를 개선하기 위해 Profile에 따른 앱 실행 속도의 특성을 파악하여 앱 실행 속도를 개선할 수 있는 Profile을 앱 개발 시점에 자동 생성하는 방법과 자동 생성한 프로파일을 APK에 포함하고 앱 설치/업데이트 시점에 활용하여 최적화를 할 수 있는 방법을 제안한다. 제안하는 방법을 통해 앱 설치 시점에 Profile에 기반하여 선택적으로 컴파일할 수 있으므로 설치 시점에 발생하는 사용자 불편을 최소화할 수 있으며 앱 설치 이후 Native Code 실행을 통해 앱 실행 속도를 최초 실행부터 개선할 수 있다.

1. 서 론

Android는 2009년 Cupcake(Android 1.5)을 시작으로 매해 새로운 기능이 포함된 새로운 버전의 Android를 공개하고 있다. 현재 최신 Android의 앱 실행 환경인 ART Runtime은 AOT(Ahead-of-time)을 지원함으로써 앱 실행 시 Native Code 실행을 통해 앱 실행 속도 및 동작 중 성능은 확보할 수 있다. 하지만 AOT 컴파일을 앱 설치 시점에 모두 수행 할 경우 설치 시간이 길어지고 컴파일로 인한 발열과 설치 이후 많은 공간을 차지하므로 인해 사용자에 불편함을 주게 된다. 이 문제를 회피하기 위해 최신 Android의 ART Runtime은 앱 설치 시점에 모두 최적화 하지 않고 설치를 마친 후 나중에 실제 사용자의 사용 이력인 Profile을 기반으로 하여 선택적인 AOT컴파일을 통해 최적화를 수행하여 앱 실행 속도 및 동작 성능을 확보하고 있다. 하지만 최신 Android(Nougat이후)에서는 AOT컴파일을 통해 최적화를 수행하기 위해서 사용자의 실제 앱 실행 이력이 있어야 하므로 언제 최적화 될지 모르는 시점까지 사용자는 앱의 성능이 최대로 나오지 않는 Interpreter 상태로 앱을 사용해야 하는 문제를 갖고 있다. 본 연구에서는 최신 Android에서 존재하는 문제를 개선하기 위하여 Profile을 앱 개발 시점에 자동 생성할 수 있는 방법과 Profile을 APK에 포함하여 앱 설치/업데이트 시점에 Profile을 활용한 최적화 방법을 제안한다.

2. Profile의 특성 파악

사용자의 사용 이력인 Profile은 사용자가 앱 실행 시 5초 이내에 사용한 Class와 실행된 Method 그리고 5초 이후에 실행되더라도 자주 실행되는 Method가 기록되어 있다. AOT컴파일 과정에서 Class는 초기화된 Image형태로 저장하여 앱 실행 시점에 Class로딩 및 초기화 시간을 단축하며 Method는 Native Code로 컴파일하여 Code수행 시간을 단축하게 된다.

하지만 사용자/앱 별 그리고 최적화 횟수 별로 앱 실행 속도를 측정해본 결과 Profile이 다름에도 비슷한 수준으로 최초 1회 최적화 이후로는 앱 실행속도가 개선되며 많은 시간 사용할수록 더 많은 Method 정보가 Profile에 기록되지만 앱 실행 속도는 더 개선되지 않는다. Class도 마찬가지이다. 그 이유로는 크게 2가지를 들 수 있다. 첫째 앱 실행 시 활용되는 Class와 Method는 앱 실행 시 첫 화면을 구성하는 UI에 연관되어 있으므로 이때 사용되는 Class와 실행되는 Method가 화면에 따라 정해져 있으며 실행 후 대기만 하는 최소 Profile로도 충분한 개선 효과를 거둘 수 있다. 둘째로는 Class와 Method정보를 더

누적하여 이를 최적화에 활용한다고 하더라도 이 결과 생성된 Class Image와 Native Code는 파일의 크기를 증가시켜 Locality가 감소함에 따라 I/O를 더 유발하고 앱 실행 속도 관점에서는 오히려 단점으로 작용할 수 있기 때문이다.

3. Profile의 자동 생성

Profile의 특성을 파악해본 결과 화면 진입만 수행한 최소 Profile만으로도 앱 실행 속도를 개선할 수 있음을 확인하였으므로 이 연구에서는 좀 더 정확한 최소 Profile을 얻기 위해 Profile Generator라는 PC기반의 Tool을 작성하였으며 이 Tool은 그림 1과 같이 ADB(Android Debug Bridge)를 통해 단말에 설치한 UIAutomator의 Agent와 Event를 주고 받으며 화면의 상태를 실시간으로 파악하여 Profile 기록을 위한 실행을 반복한다.

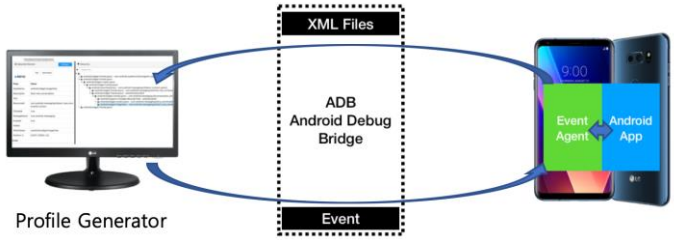


그림 1 : UIAutomator를 활용한 Profile Generator

Profile Generator의 동작은 대상 앱에 대해 화면 단위인 Activity를 실행하고 GUI 구성요소를 분석하여 1 Depth로 접근 가능한 메뉴를 탐색하고 탐색이 완료되면 Profile을 저장하고 동작을 완료한다.

4. Profile을 활용한 빠른 최적화

앱의 실행 속도를 개선할 수 있는 최소 Profile을 확보하였다면 이 Profile을 앱 설치 시점에 활용할 수 있도록 APK(Android application package)에 포함하여 배포할 수 있어야 하며 이 연구에서는 APK내에 앱에서 사용하는 파일을 저장하는 경로인 asset하위에 Profile을 저장하여 APK를 배포하고 앱 설치 시점에 Android Framework을 수정하여 Profile을 활용할 수 있도록 하였다.

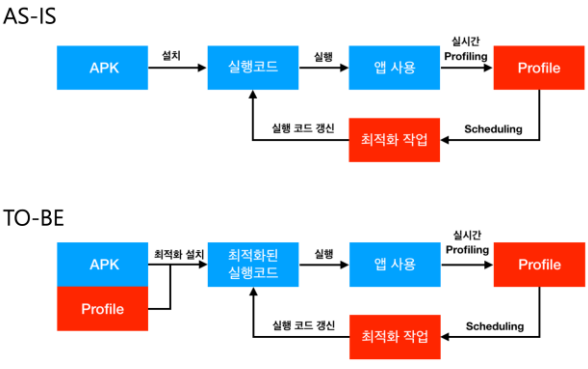


그림 2 : Profile 활용한 빠른 최적화

5. 측정

제안한 방법인 Profile Generator를 활용하여 대상 앱들에 대해 자동 실행과 자동 탐색한 결과로 얻은 Profile을 활용하여 Build/Install시점에 최적화를 수행한 테스트 결과가 표1과 같다. Contact, Music, Message, Camera, Gallery, Clock, Calendar, Settings 앱에 대해 총 100회 반복 실행한 평균이며 자동 생성한 Profile을 활용한 Build/Install시점의 최적화가 실제 사용자의 최적화 수준과 동등한 수준으로 최초 앱 실행부터 개선됨을 확인하였다.

MSM8996 +4G Android O	최적화 이전 (앱 실행 속도 비율 %)	Android 기본 최적화	Build 시점 최적화	Install시점 최적화
Contacts	100	94	95	95
Music	100	80	83	79
Message	100	96	96	95
Camera	100	95	95	95
Gallery	100	87	90	88
Clock	100	93	90	93
Calendar	100	94	96	94
Setting	100	91	91	91

표 1 : Android 기본 최적화 VS 제안 방법 적용

5. 결론 및 향후 연구

최신 Android가 지니는 문제인 최적화 되기 이전까지 앱 실행 속도가 느린 문제를 개선하기 위해 본 연구에서는 앱 사용 이력에 따라 어떤 Profile이 앱 실행 속도를 높일 수 있는 것인지를 파악하였고 앱 실행 속도를 높일 수 있는 Profile을 자동으로 생성하여 Build/Install시점에 활용함으로써 앱 실행 속도를 최초 실행부터 개선할 수 있음을 확인하였다. 이를 통해 사용자는 OOB, 앱 설치/업데이트 이후 좀 더 빠른 앱 실행 속도를 경험할 수 있을 것으로 기대한다.

향후에는 앱 실행 속도 관점 뿐만 아니라 앱 사용 중 다양한 사용 패턴에 따라 다를 수 있는 앱 동작 성능을 조기에 확보할 수 있는 방법에 대해 연구할 예정이다. 현재까지의 연구는 사용자 별로 큰 차이가 존재하지 않는 앱 실행 시점의 최적화에 초점을 맞췄고 이 목적에 부합하는 Profile을 자동확보하여 활용하는 것에 목표를 두고 진행되었기 때문에 앱 동작 중 성능은 개선이 안되기 때문이다. 이를 위해 앱 동작 성능을 높일 수 있는 Profile의 특성을 파악하고 Profile Generator의 동작을 확장하는 방법에 대한 연구가 진행된다면 앱 실행 속도 뿐만 아니라 앱 동작 성능을 조기에 확보할 수 있을 것이다.

참고문헌

[1] Youngmin Baek, Gwangui Hong, Doo-hwan Bae "A GUI State Comparison Technique for Effective Model-based Android GUI Testing

System Test 자동화 도구 제품화를 위한 품질 개선
제품 재 구조화를 통한 변경용이성 확보 사례

하승우, 백종문
LG전자, 한국과학기술원
{seungwoo.ha, jbaik} @kaist.ac.kr

Case Study of Quality Improvement for Commercialization of System Test Automation Tool

Seungwoo Ha, Jongmoon Baik
LG Electronics, Korea Advanced Institute of Science and Technology

요 약

본 연구에서는 다양한 Domain 에 적용하는 System Test 자동화 도구의 품질을 개선하고자, 구조 개선한 사례를 설명하고, 구조 개선을 통해 변경용이성을 확보 함으로서 품질이 향상 된다는 것을 검증하고자 한다. 즉 다양한 Domain 별로 적용하는 System Test 자동화 도구가 결함이 많이 나오고 있을 때, 우선적으로 구조 개선을 해야 한다. 이러한 구조 개선을 통해 2 달 기간 1 회 release 시 처리한 commit 과 resolved issues이 30% 증가 하였다. 또한 결함을 약 47% 감소 할 수 있었고, 전체 Code 중 75%를 1 일 검증 후 개별 배포를 할 수 있었다.

1. 서 론

글로벌 테스트 자동화 시장은 2016 년 30 조 에서 2025 110 조로 성장하는 것으로 예상 된다. 이러한 성장은 매년 15.4% 수준으로 성장하는 것을 예상한 시장 규모 이다.

자사는 모바일 사업부에서만 사용하고 있던 자동화 Test 도구를 TV, 자동차 부품, 가전으로 확대 전개 하였고, '16 년도부터 LG 계열사 판매를 시작 하였다. 하지만 결함이 발생된 Trend 를 보면 모바일 사업부만 사용 하였을 때 1 년에 4 건 발생 하던 결함이 전사 확산을 시작 하면서 1280 건으로 늘어 났다. 이러한 결함 발생 추이는 '17 년도 까지 크게 향상 되지 않았다.
첫 번째 원인은 모바일에서만 사용하던 도구를 다른 Domain 으로 적용 하면서 결함 수정을 할 때 마다 다른 Domain 에서 사이드 Effect로 인한 결함이 발생 하는 것이다. 또한 결함 수정 시간이 늘어 났다. 결함이 발생이 되고, 고객이 수정을 확인 하고 Close 하는 시간이 '17 년기준으로 평균 107 일이고 중앙값은 80 일이 소요 되었다.
이 중앙값에 해당하는 경우를 분석 해 보면 가장 많은 원인을 차지 하는 것이 모바일 Test Module 이다. 최초에 Mobile 사업부에서 Android 폰 대상으로 개발된 자동화 Test 도구 이다 보니, 주요 기능들이 모두 Dependency 가 걸려 있어, 작은 수정에도 모바일 Test module 을 변경 해야 하는 현상이 발생 하였다.

또한 sonarqube 를 이용한 정적 분석 결과 400 개의 결함 및 32 개 취약점이 나왔다.

두 번째 주요 원인은 Script Editor 기능 입니다.
기존에는 Eclipse 를 통하여 사용자 Script 작성 할 수 있도록 Eclipse Plugin 을 제공 하고 있었는데 이 Eclipse Plugin 이 제품과 형상이 맞지 않아서 결함이 발생 하는 현상이 일어 나고 있었다.

원인은 아래 3 가지로 요약 할 수 있다.

- Monolithic 한 Tool의 구조로 인한 Side effect
- 잠재 결함을 해소 하지 않고 제품 Release
- 별도로 관리 되는 Script Editor

이 중 이 논문에서는 Monolithic 한 Tool 의 구조로 인한 Side effect 와 잠재 결함을 해소 하는 부분만 언급 한다. (Script Editor 는 신규 Feature 개발 로 진행

하였으나, Tool 품질 개선 활동에는 포함 하지 않는다.)

2. 개선 방향

자사의 System Test 자동화 도구는 Common 영역 (공통 기능 영역) 과 Specific 영역 (특정 Domain 만 적용 하는 영역)으로 구분 하여 관리 하고 있다. Common 영역은 전체 Code 상에서 68% 를 차지 하고 있으나, Common 의 50% 영역이 신규 적용 시마다 modify 가 일어 나고 있었다.

Common 영역이 modify 가 일어나는 이유는 최초 적용 하였던 Mobile Test module 에 의존성을 가지고 있는 영역이 있어, Specific 영역이 수정이 되면 같이 수정이 일어 나는 것 이다.

의존성을 분석 하기 위해서 Mobile Test module 을 import 하고 있는 Source 의 Count 하면 아래와 같다.

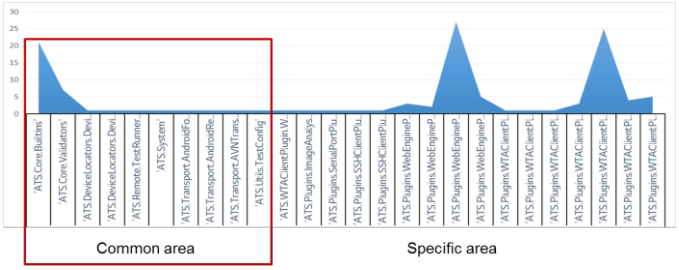


그림 1. Common 영역에서 Mobile Test Module Import Count 현황

현상) Common 영역에서 Mobile Test module 을 32 개 영역에서 Import 하고 있다.

Common 영역에서 Mobile Test module 을 import 하고 있는 주요 원인은 아래와 같다.

- Plugin 별 API 를 Core 에서 builtins 에 등록을 위한 import (Tool 에서 Script 작성 시에 사용하는 Test API 제공을 위한 기능)
- 적용 Site 별 변경 해야 하는 module이 남아 있음

결론적으로는 현재의 System test 자동화 도구는 monolithic 한 구조로 인하여 요구사항을 반영 할 때 마다 전체 Tool 을 검증이 필요했고, 이러한 검증은 최소 3 주가 소요 되었다.

개선을 위한 주요 항목은,

첫째, 적용 Site 별 변경 모듈을 별도 분리 하고 Update 하는 Plugin Arch. 구조 변경하고 Mobile Test module 을 Import 하고 있는 의존성을 제거하는 것 이다.

둘째, 정적 분석 결과로 나온 400 개 bug 와 32 개 취약점을 제거 하는 것 이다.

개선을 위한 방법은 아래와 같다

- 사이트 별 Specific 영역을 재 정의.
- Plugin based architecture 정의.
- 정의한 Architecture 기반으로 구현 및 검증.

3. 수행 내용

3.1 Tool 구조 개선 Concept

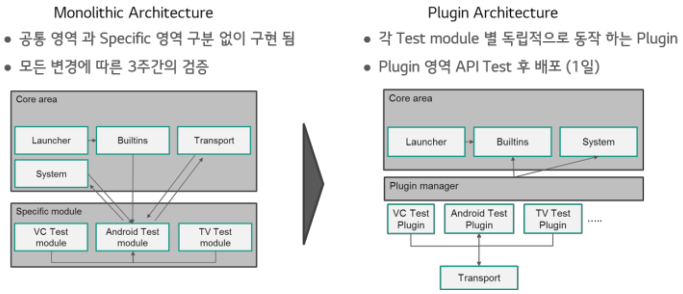


그림 2. Architecture 개선 방향

주요 내용은 Plugin architecture 를 구현 하는 것이고 각 Test module 별 독립적으로 동작하는 Plugin 개발을 목표 하고 있다. 이러한 Plugin 영역은 API Test 후 배포 가능하고, 기존 3 주간의 Test 는 1 일 검증으로 가능하게 된다.

3.2 Plugin 영역 항목 정의

UI Based Testing	Multi-media Testing	Signal Input & Measurement	Automotive Area	Protocol & Transport (Reusable)	Utility Plug-in

• Android Devices
• IOS Devices
• Web Application
• WebOS Device
• Windows
	• Audio Manager
	• Audio Analyzer
	• Image Acquision Device
	• Image Analyser
	• Image Deep Learning
	• OCR Engine
	• Video Manager
		• Constant Temperature Chamber
		• Digital Multi-Meter
		• Function Generator
		• GPS Simulator
		• Oscilloscope
		• Power Supply
		• QEMU
		• QXDM
			• Automotive Communications
			  • Interfaces (CAN, LIN, etc.)
			  • Device Vendors
			• Automotive Diagnostics
			  • OBD
			  • UDS
			• CANoe
				• Android ADB
				• Internet (TCP, UDP)
				• SSH
				• Serial Port
				• Telnet
				• VISA
					• Excel
					• (Text) Log Analyzer

5 items	7 items	8 items	4+ items	6 items	2 items

그림 3. Plugin 분류표

Specific 영역을 재 분류 하고, Plugin 화하여 필요한 Site 에만 배포 한다.

총 41 개 Plugin 으로 재 분류 하고, 각 항목별로 새로운 Plugin 개발을 목표 한다.

3.3 구현 방식

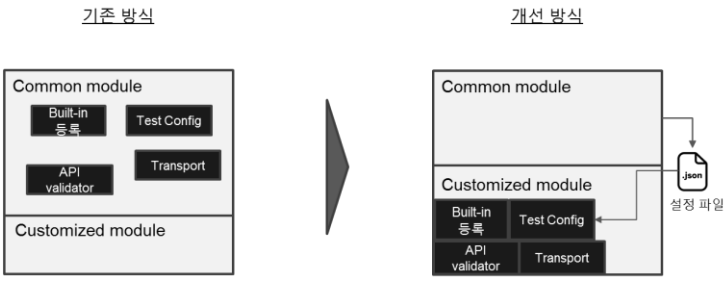


그림 4. Plugin 구현 방식

Common 영역은 변경 되지 않는 영역만 남겨 두고, 적용 대상 별 설정은 json file based 로 동적 loading 방식으로 구현 한다.

Plugin 을 관리 하기 위한 Plugin manager 를 구현 한다.

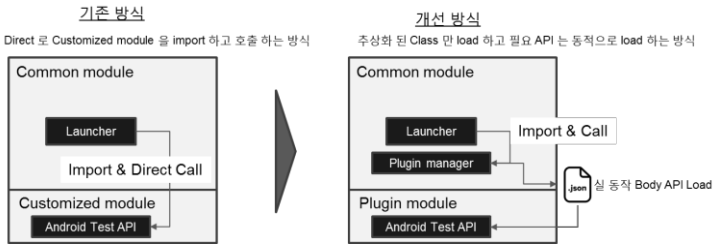


그림 5. Plugin Load 방식

Plugin 영역의 API 를 Load 할 수 있는 추상화 Class 를 정의 하고, Getattr 를 이용하여 Plugin 영역 함수를 동적으로 load 하는 방식으로 변경 하였다.

4. 수행 결과

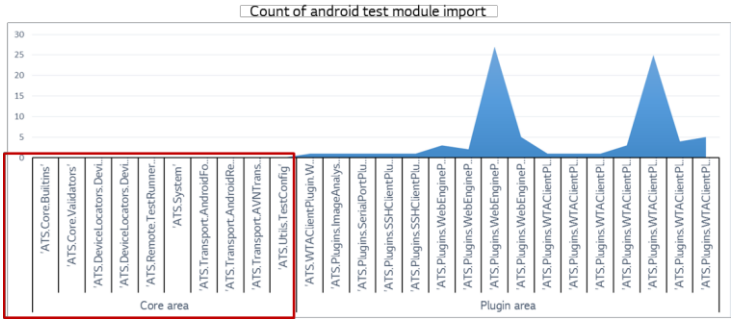


그림 6. 개선 후 Common 영역에서 Mobile Test Module Import Count 현황

Common 영역에서 android test module 을 import 하는 것이 없어 지고, 정적 분석에서 나온 결함 또한 전부 제거 하였다

또한, 기존에는 Tool Install package 를 배포 하는 방식에서 사이트 별 Plugin 만 Update 하는 방식으로 변경 하였다.

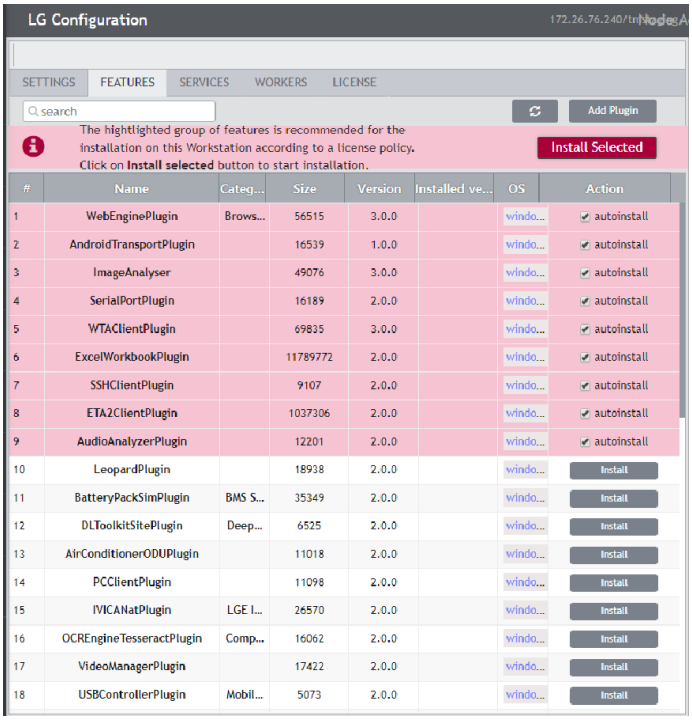


그림 7. Plugin Install/Uninstall/Update 기능 화면

대 부분 code 영역은 API Test 를 통하여 1 일 검증 하고 배포 되고, Common 영역은 전체 code 중에서 10% 정도로 줄어 들었고, 이 결과 더 빠르게 Feedback 할 수 있었다.

2 달 기간 1 회 release 시 처리 한 commit 과 resolved issues 또한 30% 증가 하였다.

5. 결론 및 향후 연구

적용 Domain 별 Specific 한 기능을 관리 할 수 있는 Plugin Architecture 를 적용 하여 품질이 향상 된 도구 확보 하였다.

적용 이후 결함을 산출 해보면 약 47% 감소 하였다.

또한, Tool 구조 개선을 통해, Code 의 75% 영역을 1 일 검증 가능하게 하였다.

향후 계획으로는 '19 년 6 월 ISO26262 인증을 받아, Automotive 영역에서 경쟁력 확보 할 예정이다.

참고 문헌

[1] Coupling and Cohesion Metrics in Java for Adaptive Reusability Risk Reduction
(seiscon 2012, M. Iyapparaja, Dr. S.Sureshkumar)

[2] From Agents and Plugins to Plugin-Agents, Concepts for Flexible Architectures
(2008 New Technologies, Mobility and Security, Benjamin Schleinzer ; Lawrence Cabac)
[3] A Plugin-Based Software Production Line Integrated Framework (2008 International Conference on Computer Science and Software Engineering, Jun Zhu ; Quan Yin ; Rui Zhu)
[4] Jupyter notook : http://jupyter.org/
[5] Transparency Market Research

Software Testing Practice

STA Global Testing Leader 테스팅컨설팅

애자일 전환을 위한 핵심요소 (Key elements for agile transformation)

본 교재는 저작권의 보호를 받고 있습니다. 저자의 승인을 받지 않은 복사, 변형, 게재 등의 일체 행위를 금지합니다.

목 차

- 애자일의 역사와 애자일 선언
  - 애자일의 역사
  - 애자일 선언
  - 애자일 원칙 및 효과
- 변화하는 요구사항
  - Stacey Model
  - 애자일 접근법
- 애자일 마인드셋
  - Doing Agile
  - Being Agile
  - 애자일 마인드셋
  - 스크럼
  - 애자일 전환의 핵심 요소

1 애자일의 역사와 애자일 선언

애자일의 역사
애자일 선언
애자일 원칙 및 효과

3

Skunk Works (1943년)

- Skunk Works - Lockheed Martin's Advanced Development Programs (ADP)
- 2차 세계대전 중 독일의 위협이 커지자 그에 대처하기 위한 전투기 개발 임무 부여
- 켈리 존슨은 직접 선발한 사람들로 팀을 구성했고, 143일 만에 XP-80 Shooting Star를 설계 및 생산함
  – F-22 개발에 거의 10년이 소요됨
- Skunk Works는 이후 U-2S sky plan, SR-71 Blackbird, F-117 스텔스기 등과 같은 혁신적인 항공기 다수를 성공적으로 개발

https://www.quora.com/How-much-time-does-it-take-to-build-a-fighter-jet-like-MiG-21-and-F-16

4

켈리 존슨의 14개 규칙 & 실천법 1

#3. 프로젝트에 관련된 사람의 수를 거의 악의적이라고 볼 수 있을 정도로 제한해야 한다. 소수의 좋은 사람만을 활용해라. (일반적인 시스템 대비 10%~25%만 참여)

#4. 이해하기 쉬운 도면과 변경 사항을 쉽게 적용할 수 있는 유연한 도면 릴리즈 시스템이 제공되어야 한다.

#5. 필요한 보고서 수는 최소로 유지되어야 하지만, 주요 작업은 철저하게 기록되어야 한다.

#12. 군사 프로젝트 조직과 계약사 간의 상호 신뢰가 있어야 하며, 매일 이루어지는 긴밀한 협업과 소통이 필요하다. 이렇게 함으로써, 오해나 불필요한 서신 교환을 최소한으로 줄일 수 있다.

5 STA

켈리 존슨의 14개 규칙 & 실천법 1

#3. 프로젝트에 관련된 사람의 수를 거의 악의적이라고 볼 수 있을 정도로 제한해야 한다. 소수의 좋은 사람만을 활용해라. (일반적인 시스템 대비 10%~25%만 참여)

#4. 이해하기 쉬운 도면과 변경 사항을 쉽게 적용할 수 있는 유연한 도면 릴리즈 시스템이 제공되어야 한다.

#5. 필요한 보고서 수는 최소로 유지되어야 하지만, 주요 작업은 철저하게 기록되어야 한다.

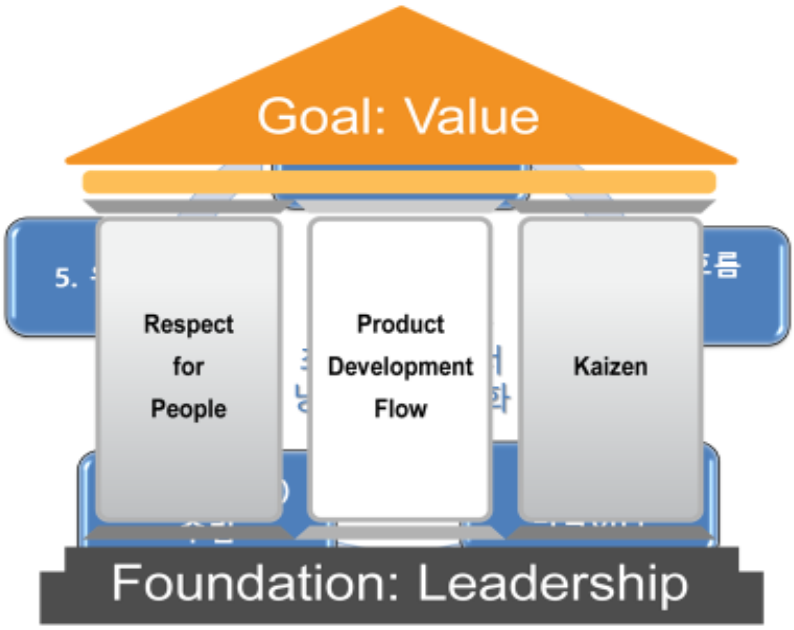
#12. 군사 프로젝트 조직과 계약사 간의 상호 신뢰가 있어야 하며, 매일 이루어지는 긴밀한 협업과 소통이 필요하다. 이렇게 함으로써, 오해나 불필요한 서신 교환을 최소한으로 줄일 수 있다.

5 STA

린(lean) 소프트웨어 개발 - 1993년/2003년

1

- 7가지 원리
  - 전체 최적화
  - 낭비 최소화
  - 품질 내재화
  - 지속적인 학습
  - 빠른 인도
  - 모두를 참여시킴
  - 끊임없는 개선



Robert Charette, 1993, 최조 제안
7 Principles form 2003, Mary and Tom Poppendieck

SCALED AGILE

7

STA

익스트림 프로그래밍 (XP, eXtreme Programming) - 1996년

1

- 켄트 벡(Kent Beck)이 크라이슬러에서 실험적인 소프트웨어 개발 프로젝트에 최초 적용
  - 프로젝트는 실패했지만, 아이디어는 이후 발전
- 켄트 벡의 기본적인 생각
  - 이미 알려진 좋은 실천법(practice)을 극한까지 수행하자.

좋은 실천법	XP 실천법
짧은 반복주기	플래닝 게임
소프트웨어 설계	지속적 리팩토링
단순성	제대로 동작하는 최소한
코드 리뷰	짝 프로그래밍
테스팅	테스트 주도 개발
통합 테스팅	지속적 통합

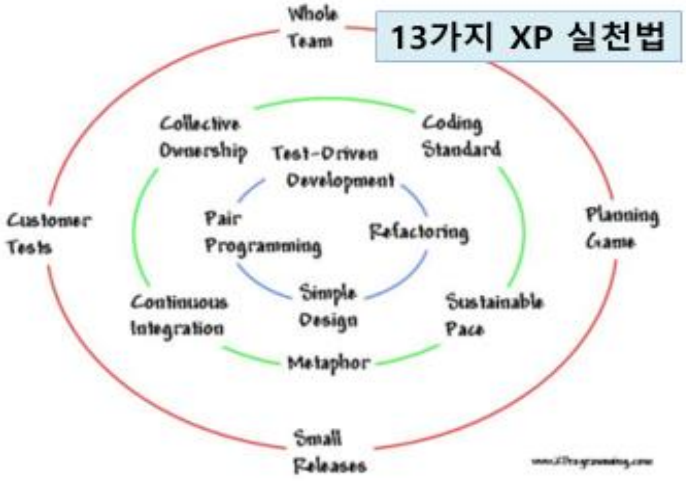
8

STA

XP 개요

1

- **1~4주 반복주기**
  - 가장 중요한 스토리를 먼저 개발
  - 30~120일 릴리즈 주기
- **개발자(programmer) 주도**
- **작은 팀**
- **스스로 관리하는 팀**
- **기본적인 원리:**
  - 단순화
  - 점진적 변화
  - 변화 수용
  - 좋은 품질의 작업



9

STA

애자일 선언 - 2001년

1

"애자일에서 우리는

Manifesto for Agile Software Development

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

That is, while there is value in the items on the right, we value the items on the left more.

변화에 대응하기를

이 말은, 왼쪽에 있는 것들도 가치가 있지만,
우리는 오른쪽에 있는 것들에 더 높은 가치를 둔다는 것이다."

10

STA

애자일 원칙 (1/2)

1

- 우리의 최우선 순위는, 가치 있는 소프트웨어를 일찍 그리고 지속적으로 전달해서 고객을 만족시키는 것이다.
- 비록 개발의 후반부일지라도 요구사항 변경을 환영하라. 애자일 프로세스들은 변화를 활용해 고객의 경쟁력에 도움이 되게 한다.
- 작동하는 소프트웨어를 자주 전달하라. 두어 주에서 두어 개월의 간격으로 하되 더 짧은 기간을 선호하라.
- 비즈니스 쪽의 사람들과 개발자들은 프로젝트 전체에 걸쳐 날마다 함께 일해야 한다.
- 동기가 부여된 개인들 중심으로 프로젝트를 구성하라. 그들이 필요로 하는 환경과 지원을 주고 그들이 일을 끝내리라고 신뢰하라.

11 STA

애자일 원칙 (2/2)

1

- 개발팀으로, 또 개발팀 내부에서 정보를 전하는 가장 효율적이고 효과적인 방법은 면대면 대화이다.
- 작동하는 소프트웨어가 진척의 주된 척도이다.
- 애자일 프로세스들은 지속 가능한 개발을 장려한다. 스폰서, 개발자, 사용자는 일정한 속도를 계속 유지할 수 있어야 한다.
- 기술적 탁월성과 좋은 설계에 대한 지속적 관심이 기민함을 높인다.
- 단순성이 - 안 하는 일의 양을 최대화하는 기술이 - 필수적이다.
- 최고의 아키텍처, 요구사항, 설계는 자기 조직적인 팀에서 창발한다.
- 팀은 정기적으로 어떻게 더 효과적이 될지 숙고하고, 이에 따라 팀의 행동을 조율하고 조정한다.

12 STA

애자일의 효과 1

Scaled Agile Framework SCALED AGILE 13 STA

애자일의 효과 1

Top 10 Benefits of Agile (12th State of Agile Survey, 2018)

- 변화하는 우선순위 관리 가능
- 프로젝트 가시성
- 비즈니스/IT 협업
- 릴리즈 속도/타임 투 마켓 (time-to-market)
- 향상된 팀 생산성
- 팀 사기 증진
- 프로젝트 예측 가능성
- 소프트웨어 품질
- 프로젝트 리스크 완화
- 엔지니어링 규율(discipline)
- 분산된 팀 관리
- 소프트웨어 유지보수 가능성
- 프로젝트 비용 인하

5대 효과 (HP Survey, 2016/17)

- 협력 향상
- 소프트웨어 품질 향상
- 고객 만족 향상
- 빨라진 타임 투 마켓
- 개발 비용 절감

14 STA

IT 프로젝트 성공/실패 비율

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	20%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Standish Project Database, 2011-15

15

STA

애자일 채택률

- **Primary Development Method (HP Survey, 2016/17)**
  - 애자일에 가까운 (51%)
  - 하이브리드 (24%)
  - 순수 애자일 (16%)
  - 폭포수에 가까운 (7%)
  - 순수 폭포수 (2%)

- **정부 지원 / 강제**
  - 미국
  - 영국
  - 호주
  - 홍콩
  - 싱가폴
  - *대한민국*

16

STA

2

변화하는 요구사항

Stacey Model

애자일 접근법

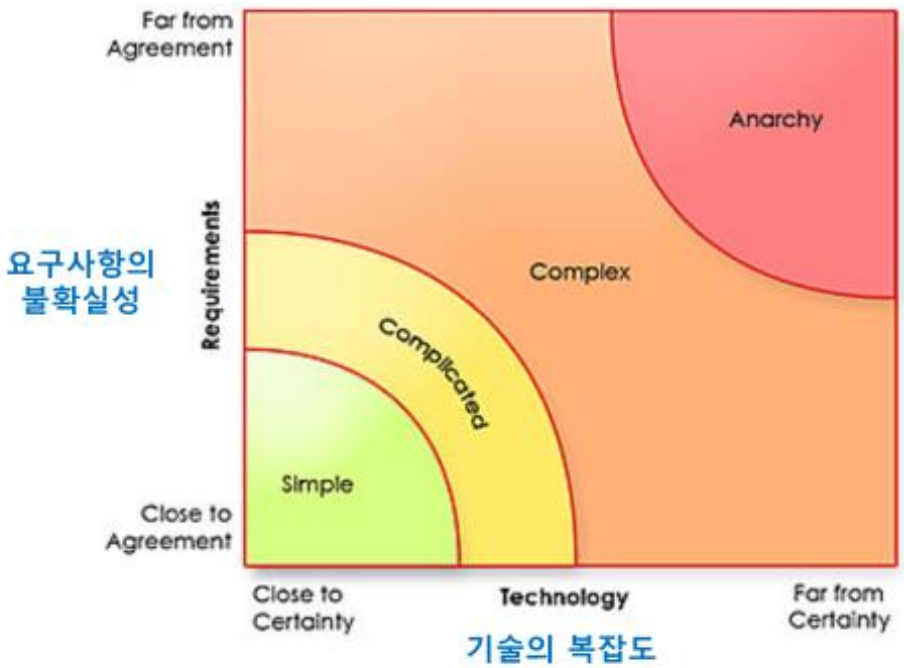
17

STA

Stacey의 복잡도(Complexity) 모델

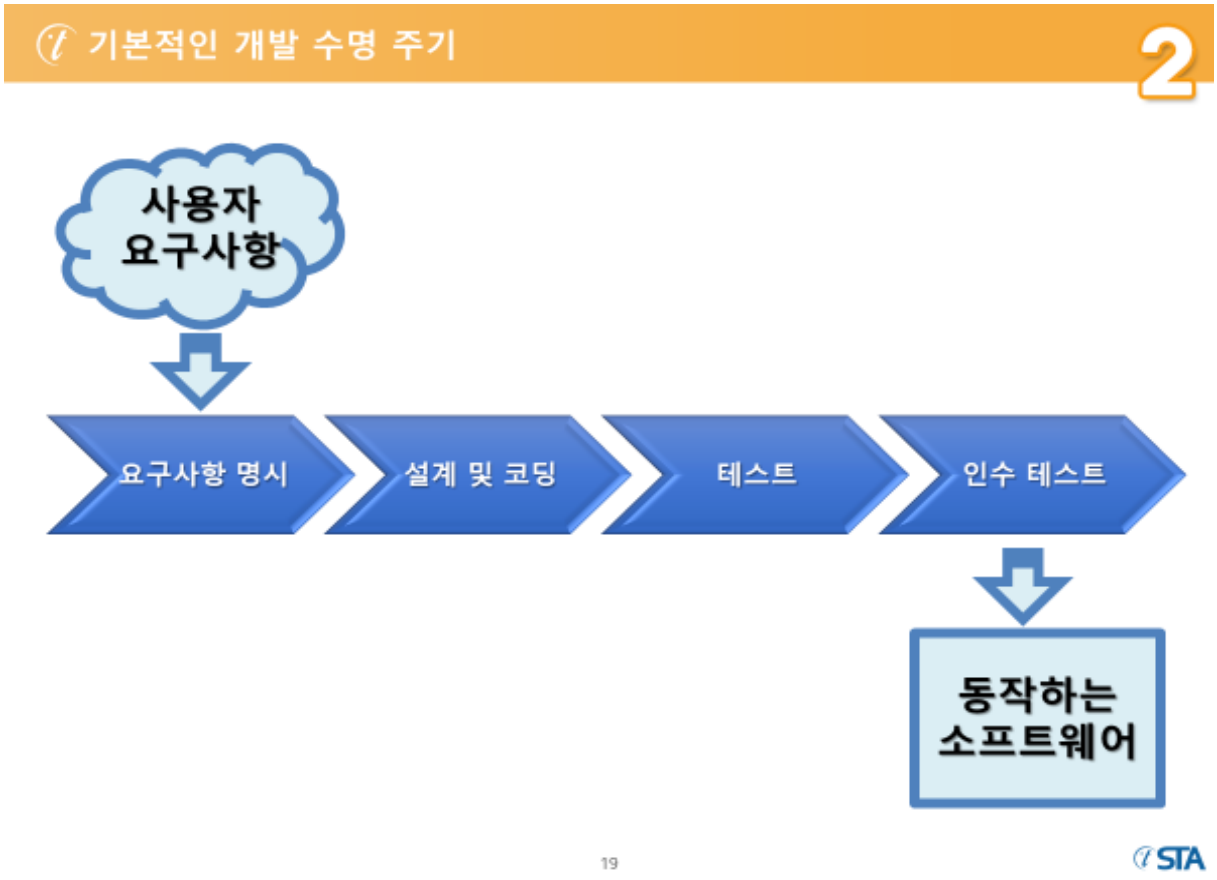
2

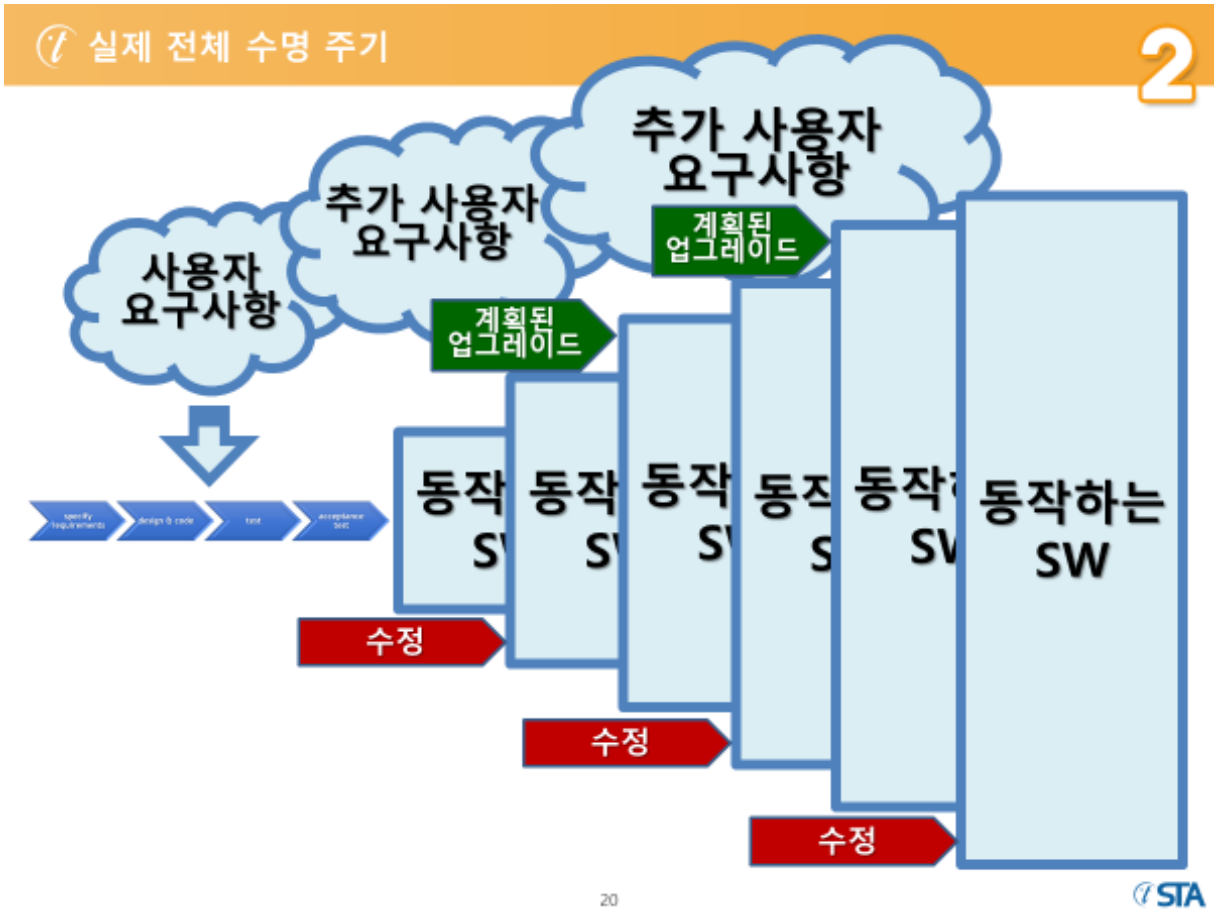
- 여러분의 프로젝트는 어디에 속합니까?



18

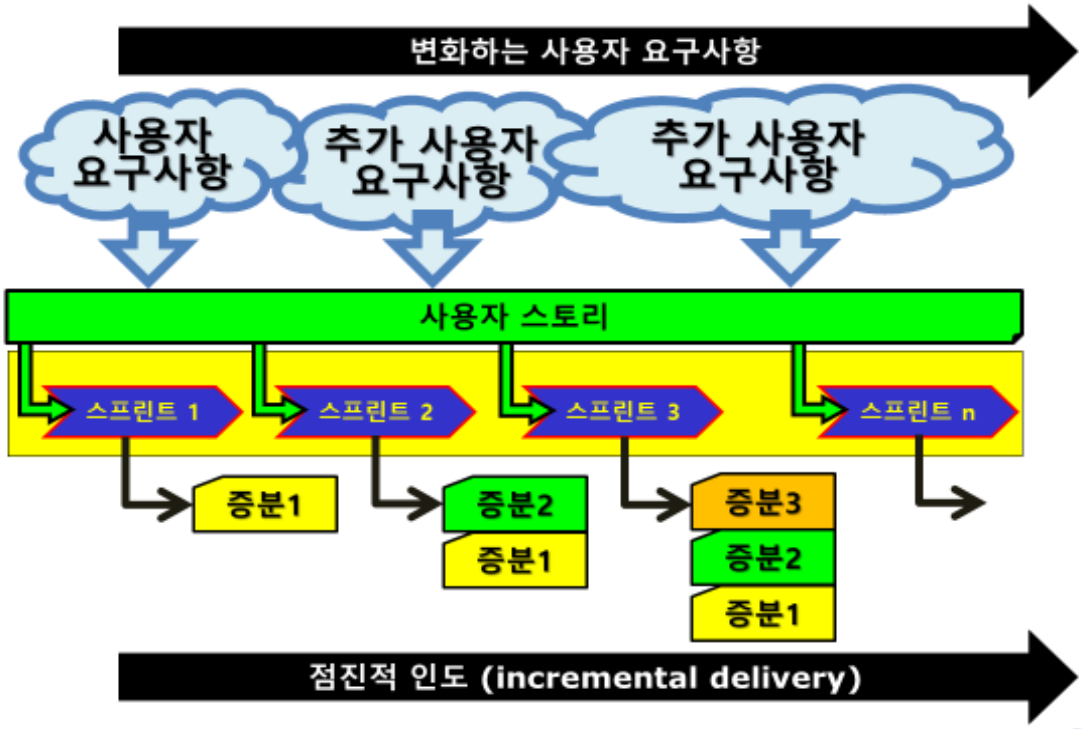
STA


기본적인 개발 수명 주기
2
사용자
요구사항
요구사항 명시
설계 및 코딩
테스트
인수 테스트
동작하는
소프트웨어
19
STA


실제 전체 수명 주기
2
사용자
요구사항
추가 사용자
요구사항
추가 사용자
요구사항
계획된
업그레이드
계획된
업그레이드
동작
SW
동작
SW
동작
SW
동작
SW
동작
SW
동작하는
SW
수정
수정
수정
20
STA

애자일 점전직 인도

2

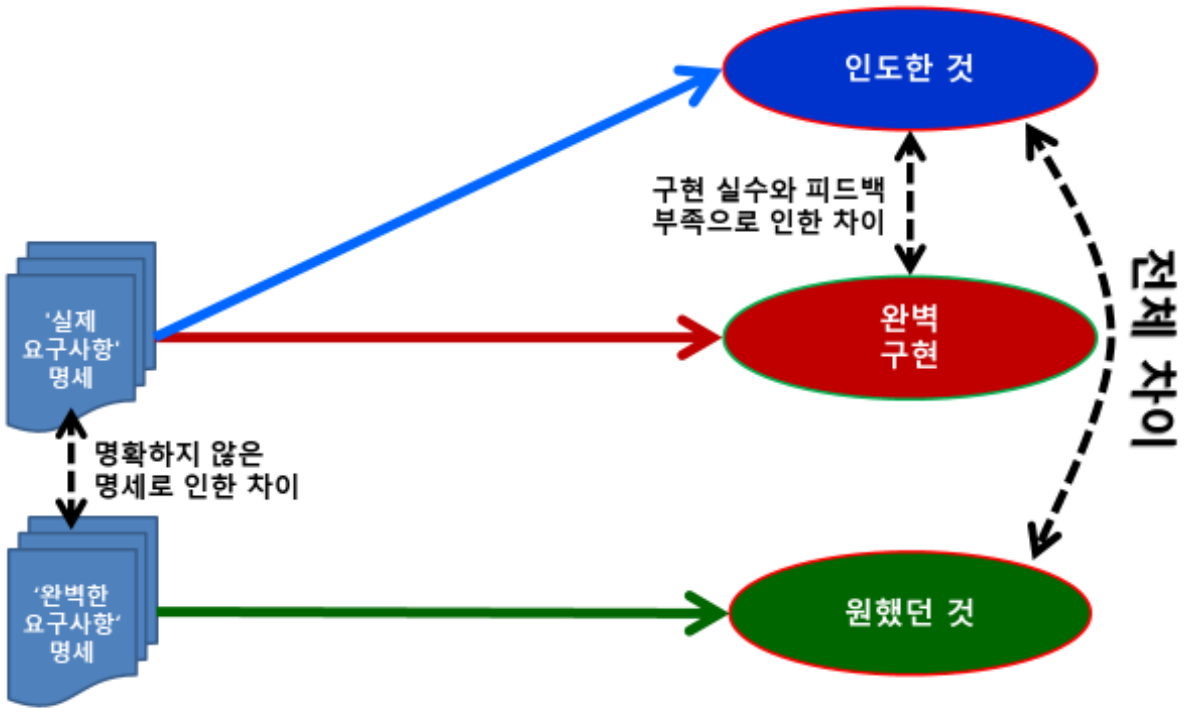


21

STA

전통적인 접근법

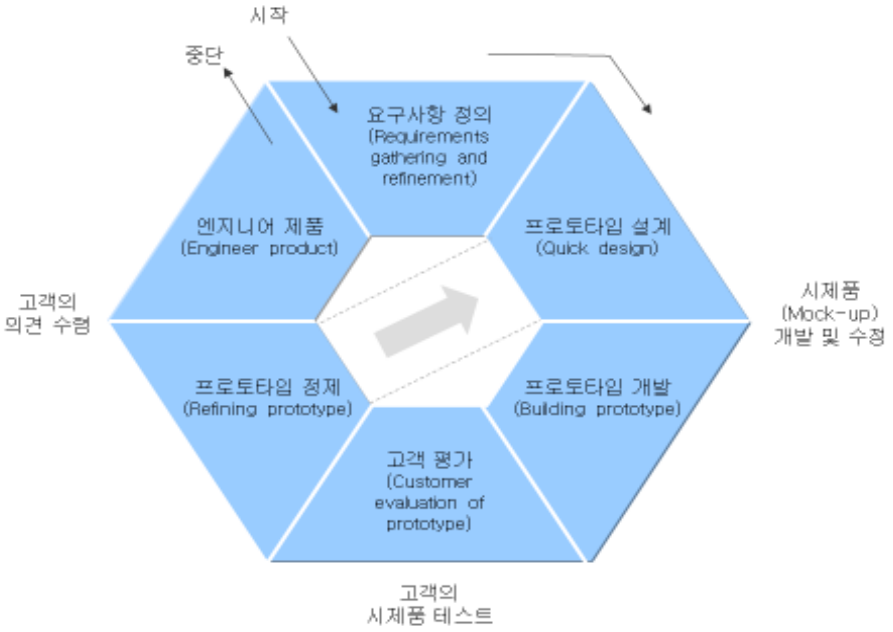
2



22

STA

프로토타입 모델 (Prototyping Model) 2

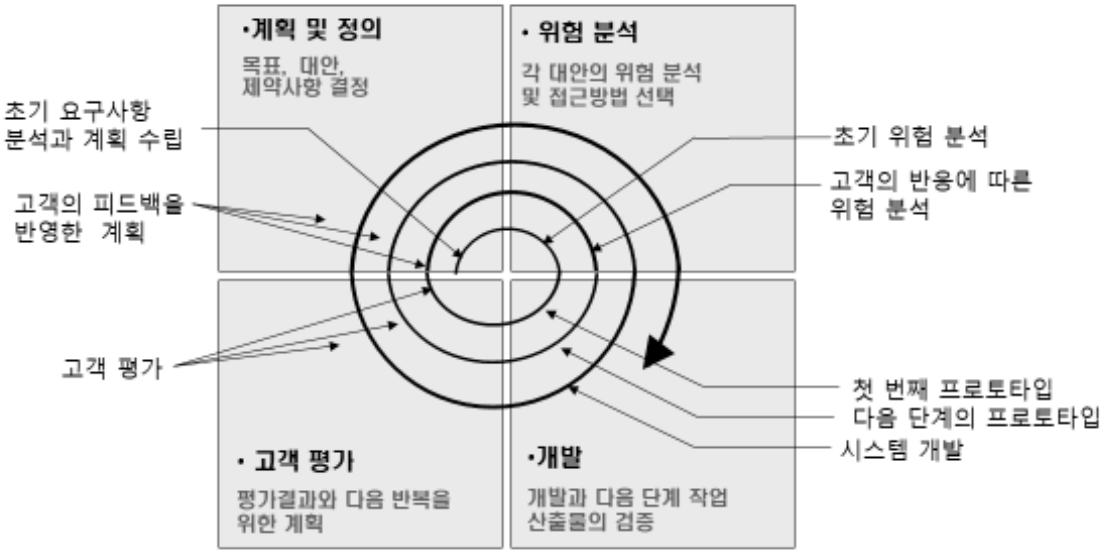


23

STA

나선형 모델(Spiral Model) 2

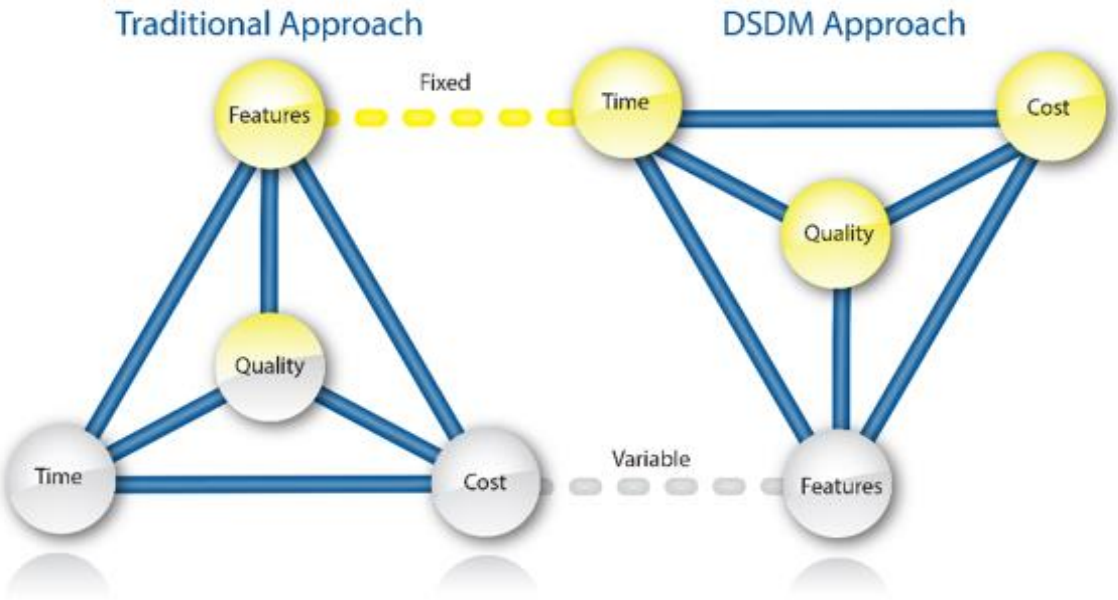
- 폭포수 모형과 프로토타입 모형의 장점을 수용, 위험 분석(Risk analysis)을 추가한 점증적 개발 모델
- 프로젝트 수행 시 발생하는 위험을 관리하고 최소화 하려는 것이 목적



24

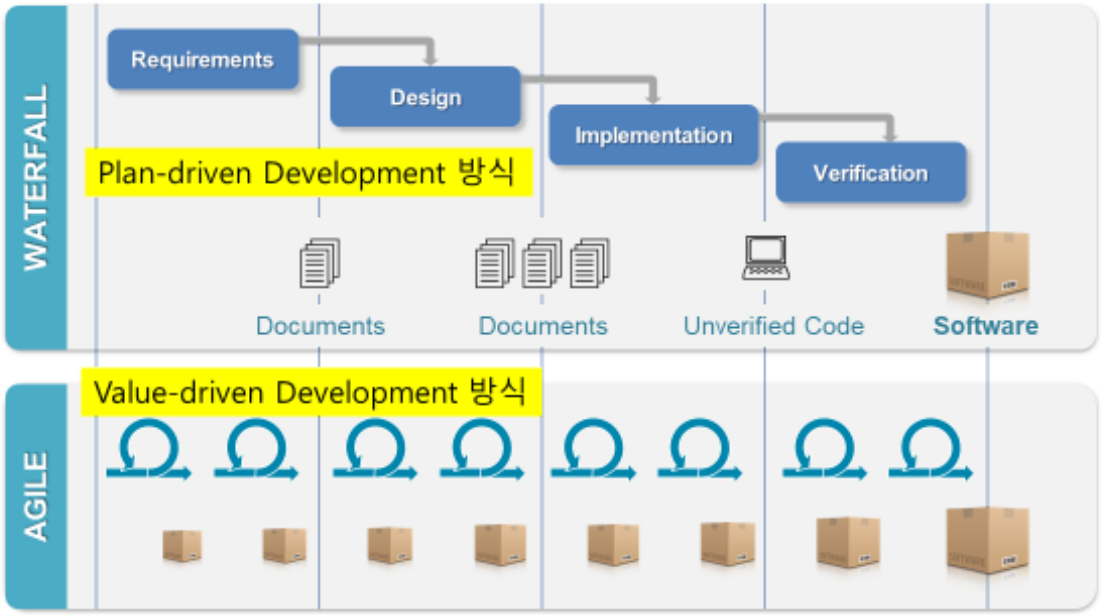
STA

애자일 접근법 2



25 STA

전통적인 접근법 VS 애자일 접근법 2



Scaled Agile Framework

26 STA

애자일 마인드셋

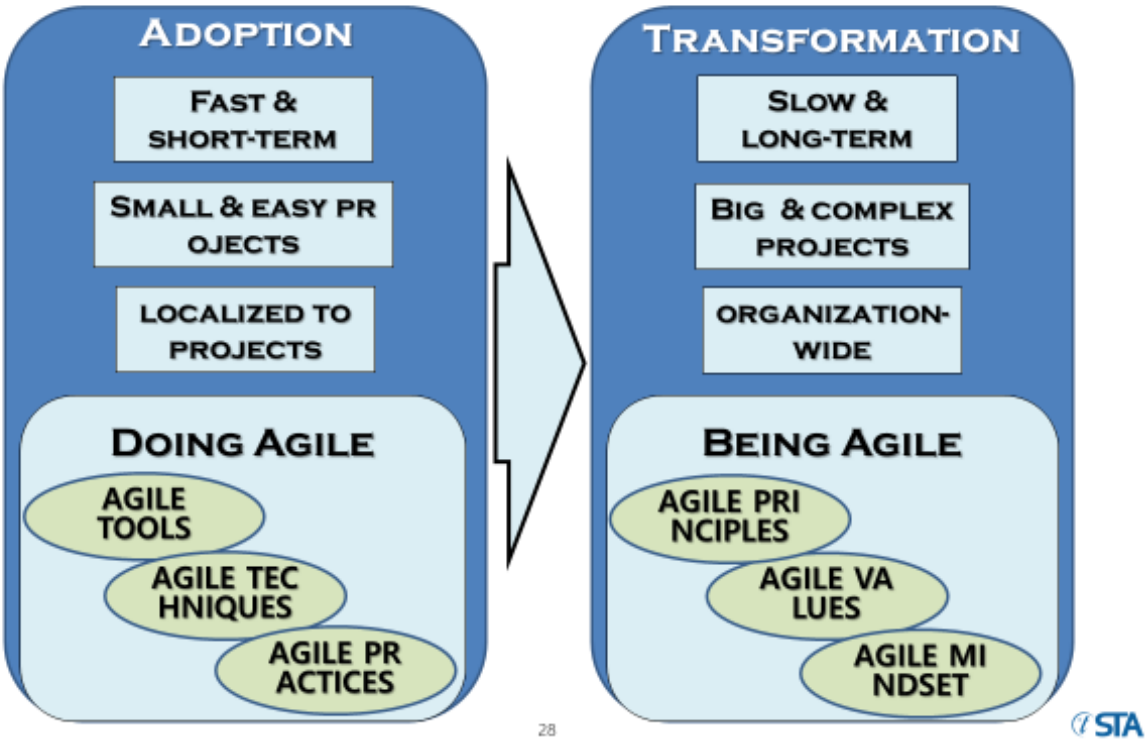
Doing Agile
Being Agile
애자일 마인드셋
스크럼
애자일 전환을 위한 핵심 요소

27

STA

Readiness for Adoption or Transformation?

3



28

STA

Doing Agile - Agile Tools
12th State of Agile Survey, 2018

3

Unit Testing	75%
Coding Standards	64%
Continuous Integration	54%
Refactoring	45%
Continuous Deployment	37%
Pair Programming	36%
Test-Driven Development (TDD)	35%
Automated Acceptance Testing	32%

29

STA

Being Agile

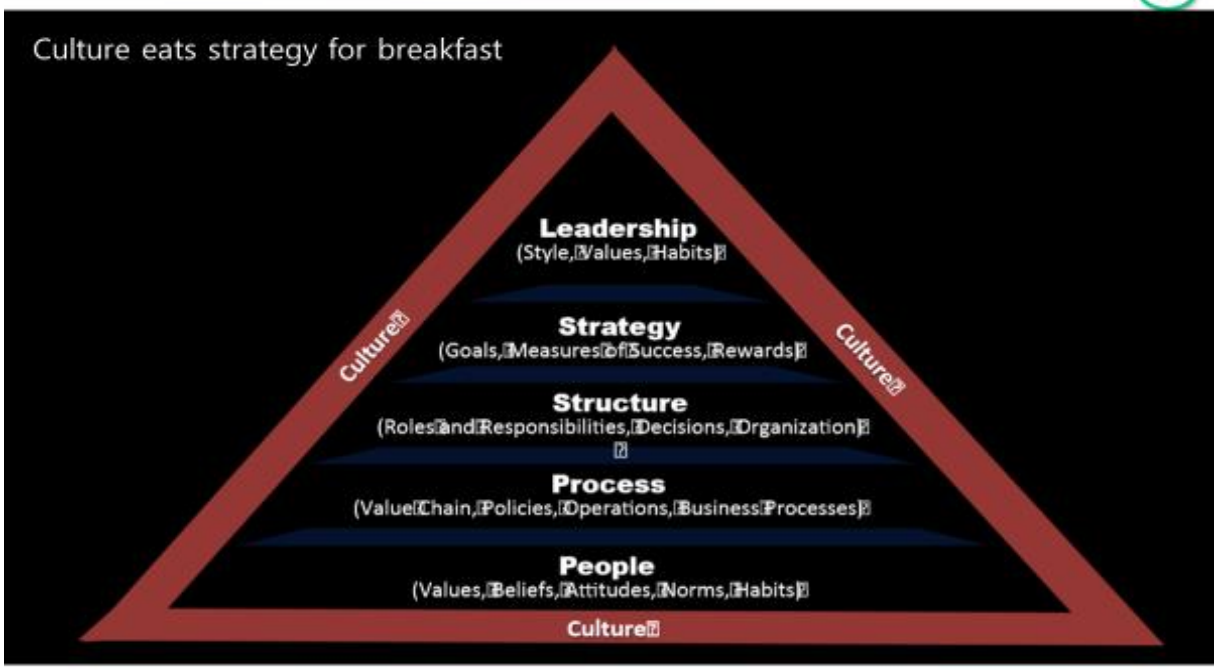
3

- **12 Agile Principles**
  - See Agile Manifesto
- **Agile Values**
  - Trust, Respect & Courage
- **Agile Mindset**
  - Being Positive
  - Embracing Failure
  - User-Centric
  - Team-Driven
  - Expecting to Improve
  - Adaptability

30

STA

조직의 문화 3



Sidky / Riot Games
Mindset & Culture: At the Heart of a Winning Agile Transformation Ahmed

31 STA

Agile 도입이 성공하려면? 3

- 애자일 도입이란 무엇인가?
- 방법론 적용?
  - 스크럼을 도입하면 애자일인가?
  - XP를 하면 애자일인가?
  - CI를 구축하면 애자일인가?
  - TDD를 하면 애자일인가?

- 신뢰 & 긍정적 태도 (Trust & Being Positive)
- 실패 수용 (Embracing Failure)
- 사용자 중심 (User-Centric)
- 팀 주도 (Team-Driven)
- 개선에 대한 기대감 (Expecting to Improve)
- 변화에 대응 (Adaptability)

NO!!!

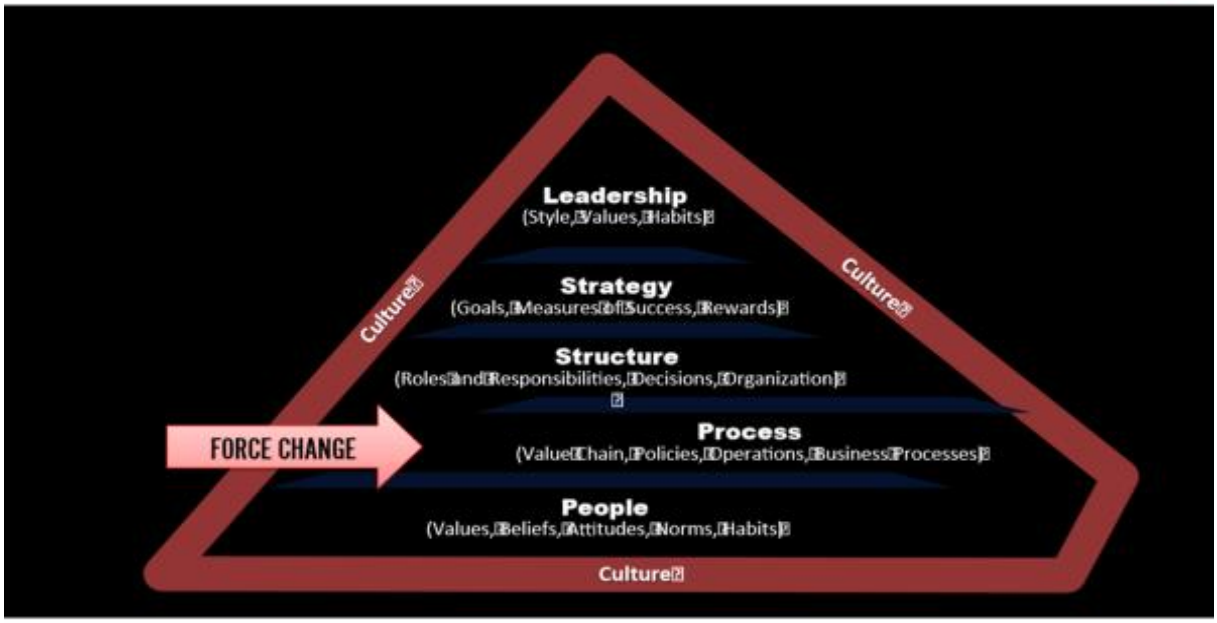
'애자일을 도입했다.' 또는 '애자일하게 개발하고 있다." 는 것은
'애자일 선언과 원칙을 따르기 위해 얼마나 노력하고 있는가?'이다.

조직의 문화를 바꿔야 한다!!

32 STA

애자일 마인드셋 – 프로세스 주도로 변화할 경우

3



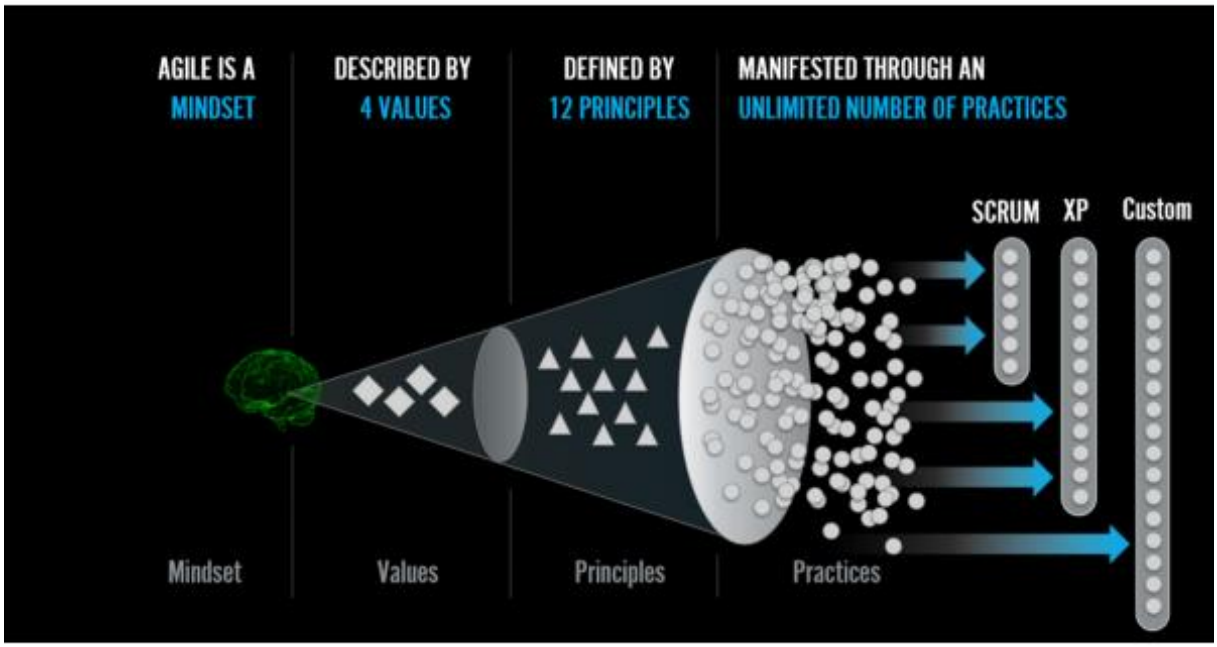
Sidky / Riot Games
Mindset & Culture: At the Heart of a Winning Agile TransformationAhmed

33

STA

애자일 마인드셋 – 성장 마인드셋

3

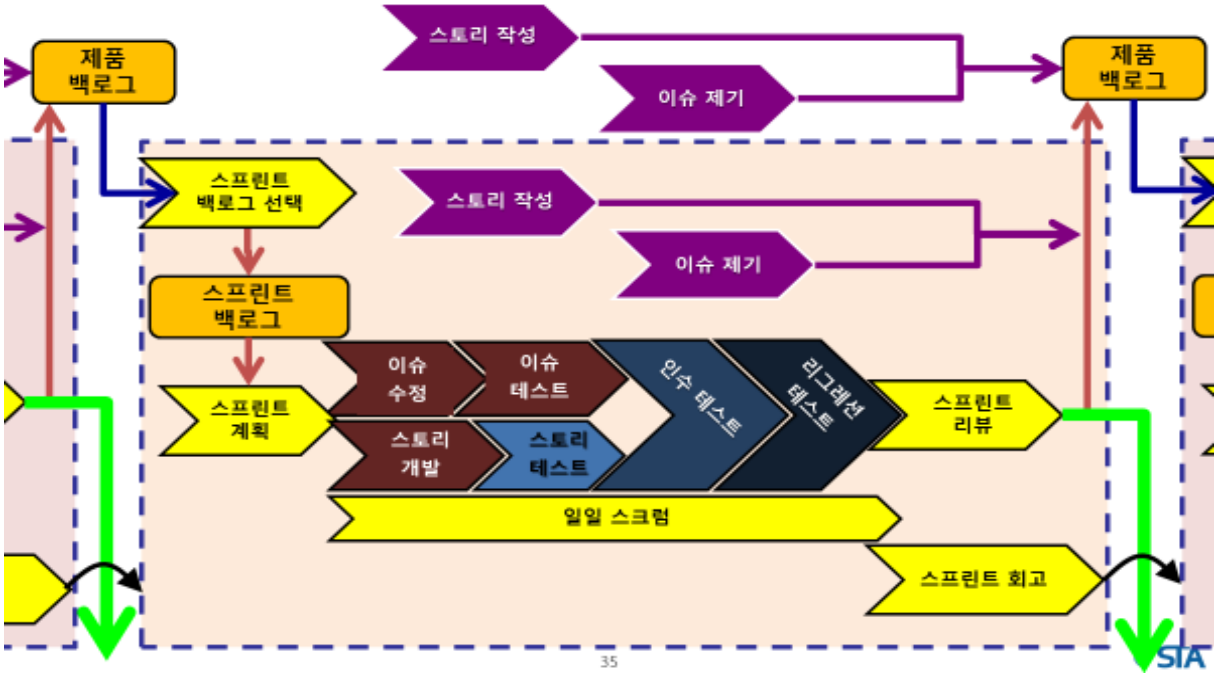


Sidky / Riot Games
Mindset & Culture: At the Heart of a Winning Agile TransformationAhmed

34

STA

스크럼 프로세스



애자일 전환 – 프로세스 주도(1)

Sidky / F
TransformationAhmed

36

STA

애자일 전환을 위한 핵심 요소! 3

- **Agile Adoption is only the first step in an Agile Transformation**
- **The top two challenges are:**

Organizational Culture

Management Commitment

– Moving to agile is NOT a technical challenge

37

STA

감사합니다.

www.sten.or.kr / www.sta.co.kr

STA

Testing Artificial Intelligence Systems

본 자료는 저작권의 보호를 받고 있습니다. 저자의 승인을 받지 않은 복사, 변형, 게재 등의 일체 행위를 금지합니다.
STA테스팅컨설팅

Definition used for Autonomous System

- **Autonomy**
  - the capacity to make an informed, un-coerced decision. Autonomous organizations or institutions are independent or self-governing
  - the ability to act independently of direct human control and in unrehearsed conditions
- **Autonomous System**
  - 'smart' or 'intelligent' system that is aware of, and interacts with, its environment to independently achieve delegated objectives

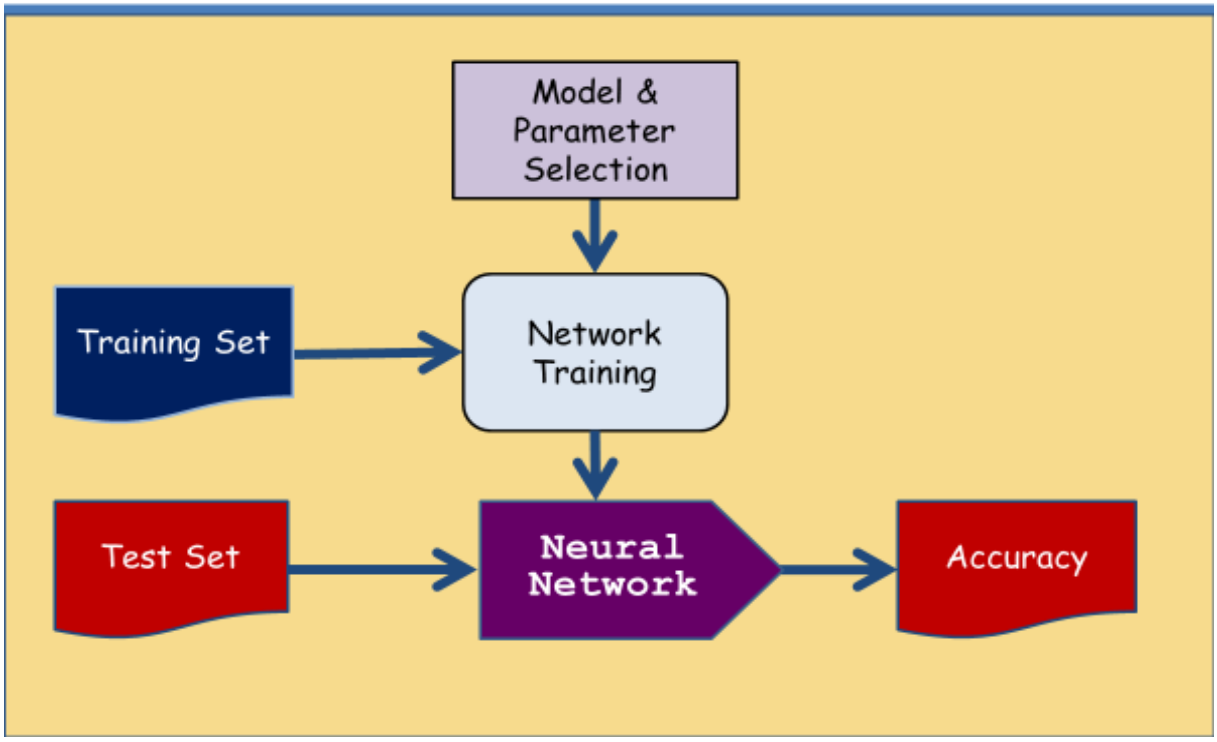
Examples: Autonomous Systems

- **Autonomous systems can be physical or purely digital**
- **Transportation**
  - Cars / Trucks
  - Unmanned Aircraft (Drones)
  - Ships/Boats
  - Trains
- **Robotic/IoT Platforms**
  - E.g. manufacturing, vacuum cleaners, smart thermostat
- **Medical Diagnostics**
- **Smart Buildings / Cities / Energy / Utilities**
- **Financial Systems**
  - E.g. automated market trading system

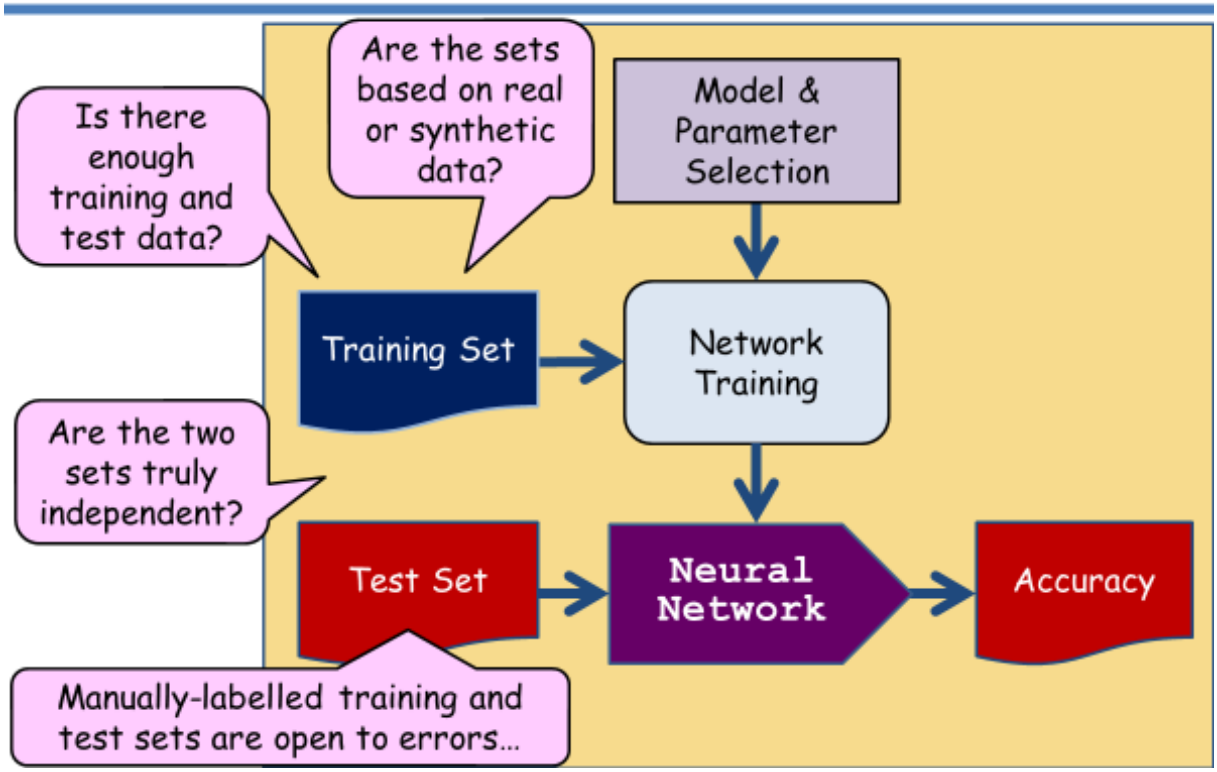
Failures: Autonomous Systems

- **Accidents**
  - Tesla, 2016
    - Joshua Brown died in a Tesla Model S
    - On a bright day, car failed to spot a white 18-wheel truck/trailer
  - Uber, 2018
    - Elaine Herzberg killed crossing the street with her bicycle in Arizona by a car travelling at 38 mph
    - 10 pm at night, but visible for about two seconds before the collision
  - Überlingen mid-air collision accident
    - Manufacturer of collision avoidance system (TCAS) installed on board the aircraft involved held responsible
- **Flash Crashes on Stock Markets**
  - Extreme flash crashes can send stock markets tumbling in minutes
  - Regulators have managed these risks by installing circuit breakers
- **Google searches were showing high-paying jobs only to male users**
- **AI-based policing systems in the US biased against African Americans**

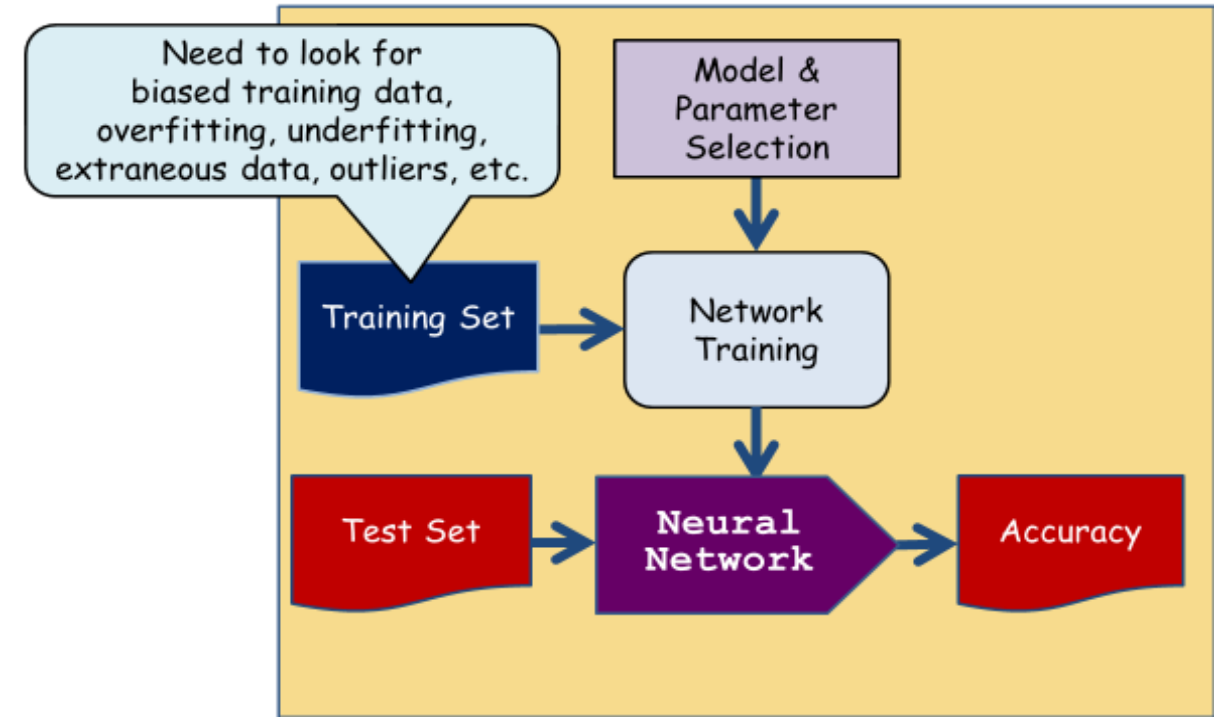
Supervised Machine Learning



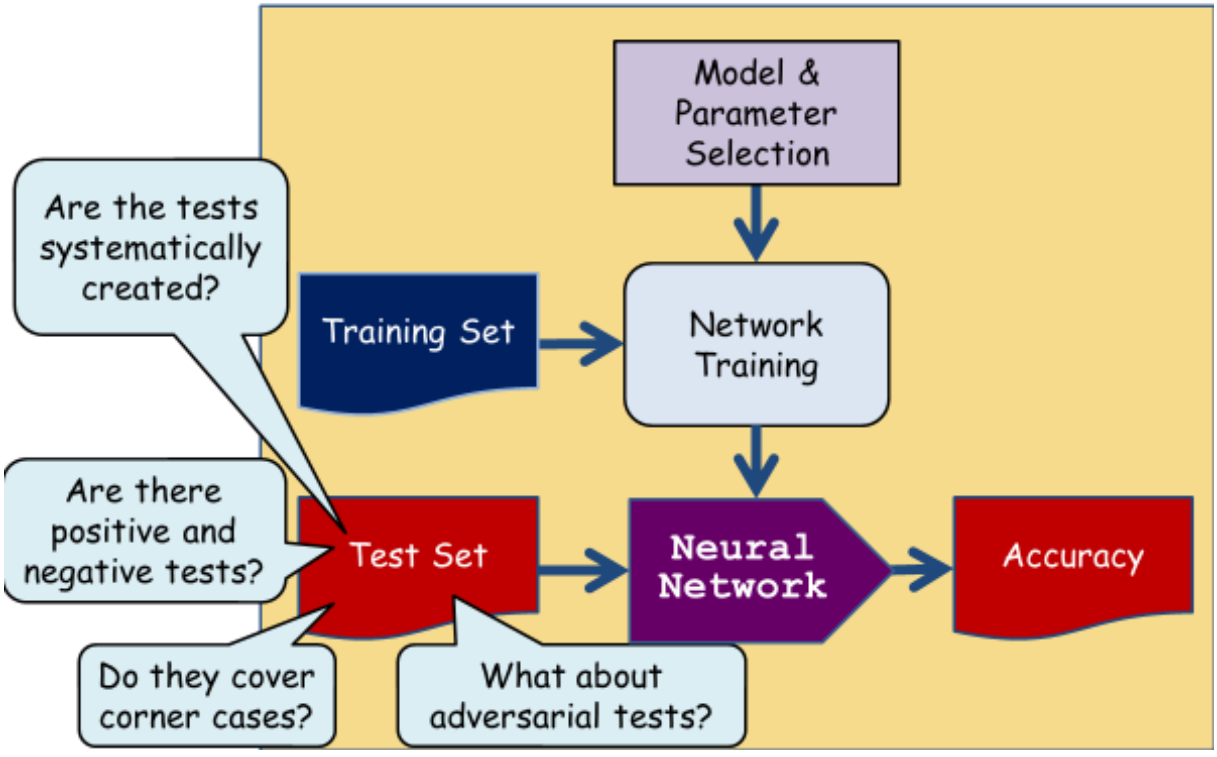
Checking the Training & Test Sets



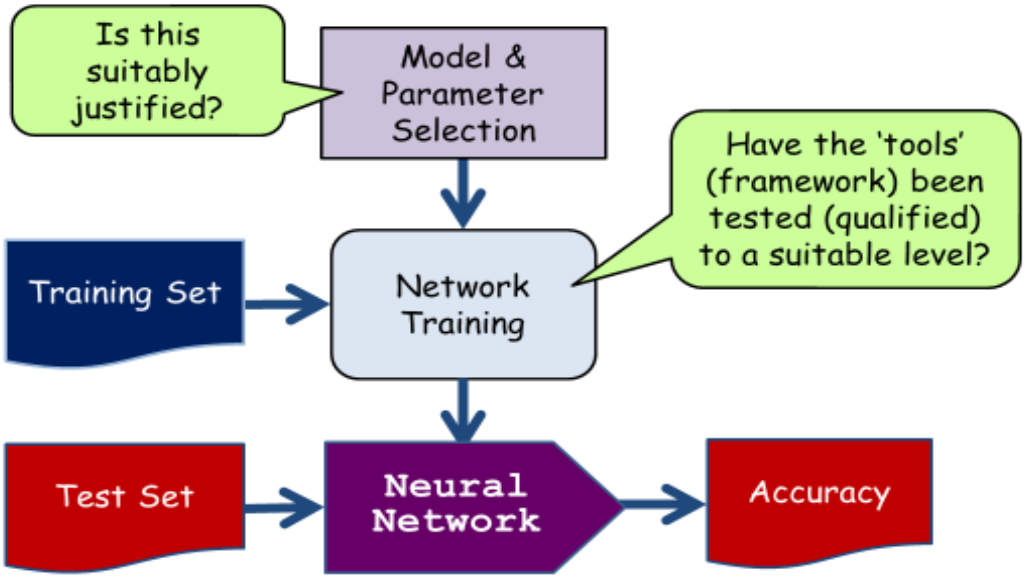
Checking the Training Set



Checking the Test Set



Checking the Training

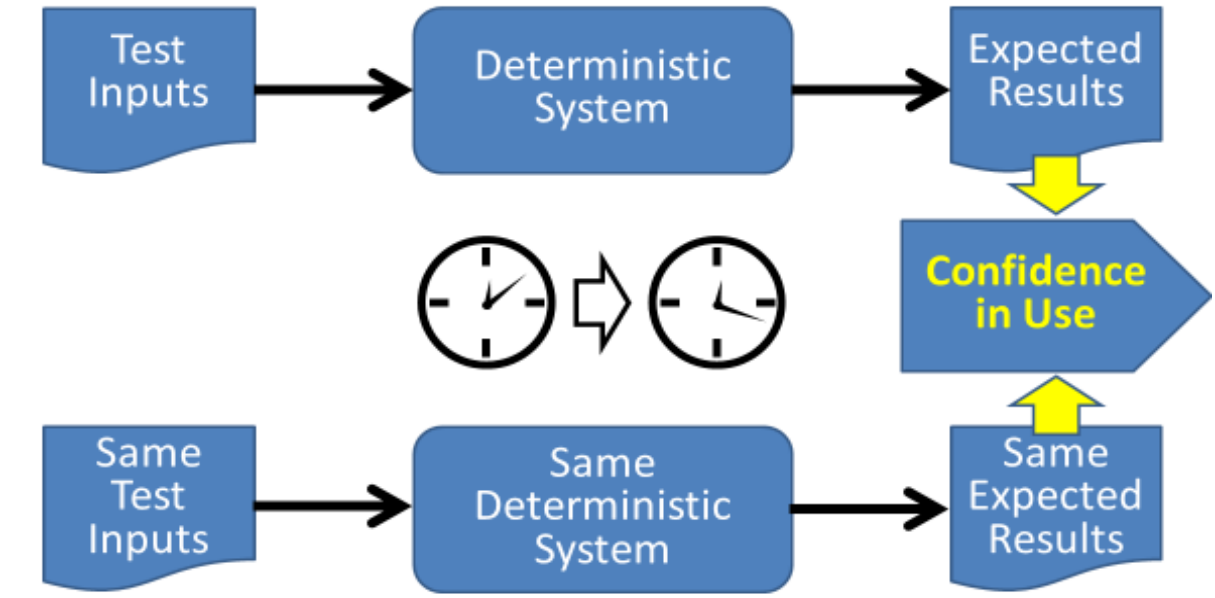


.

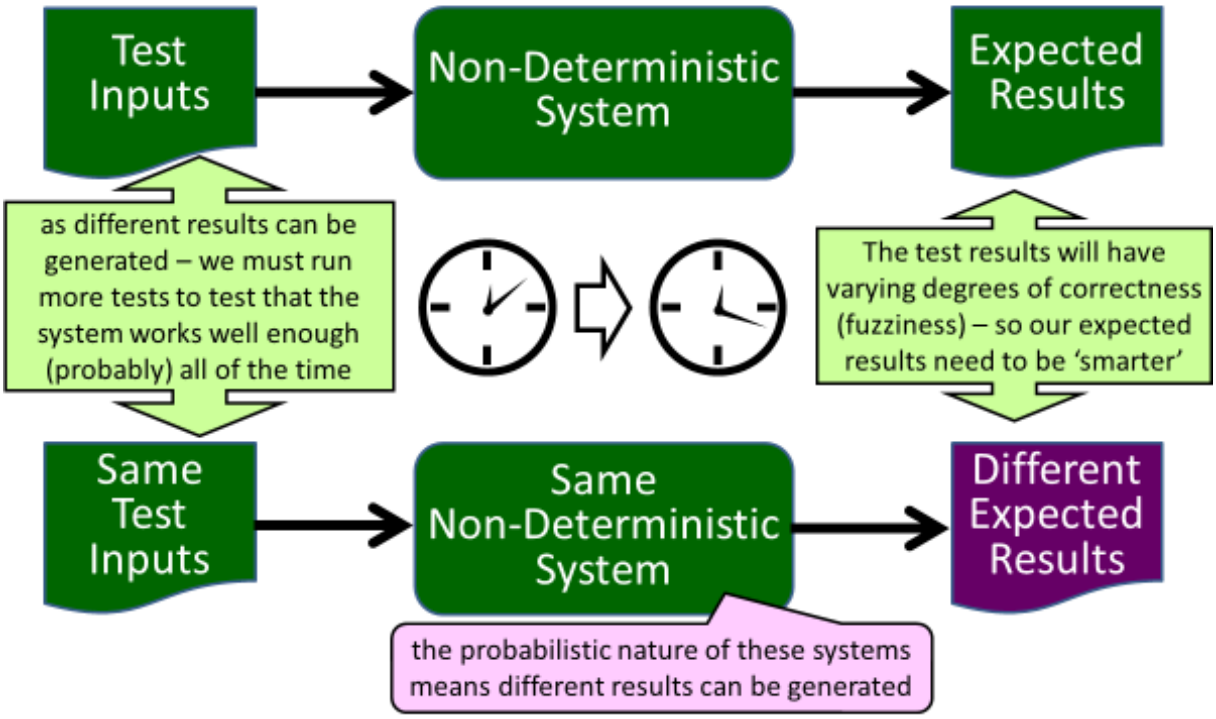
Machine Learning Challenges for Safety

- **Lack of high-integrity**
  - Machine Learning Frameworks
  - Machine Learning Software Libraries
- **Lack of standards covering the development of high-integrity autonomous systems**
- **Poor understanding of how interacting autonomous systems may cause failures**
  - e.g. interacting market trading systems causing flash crashes on stock markets

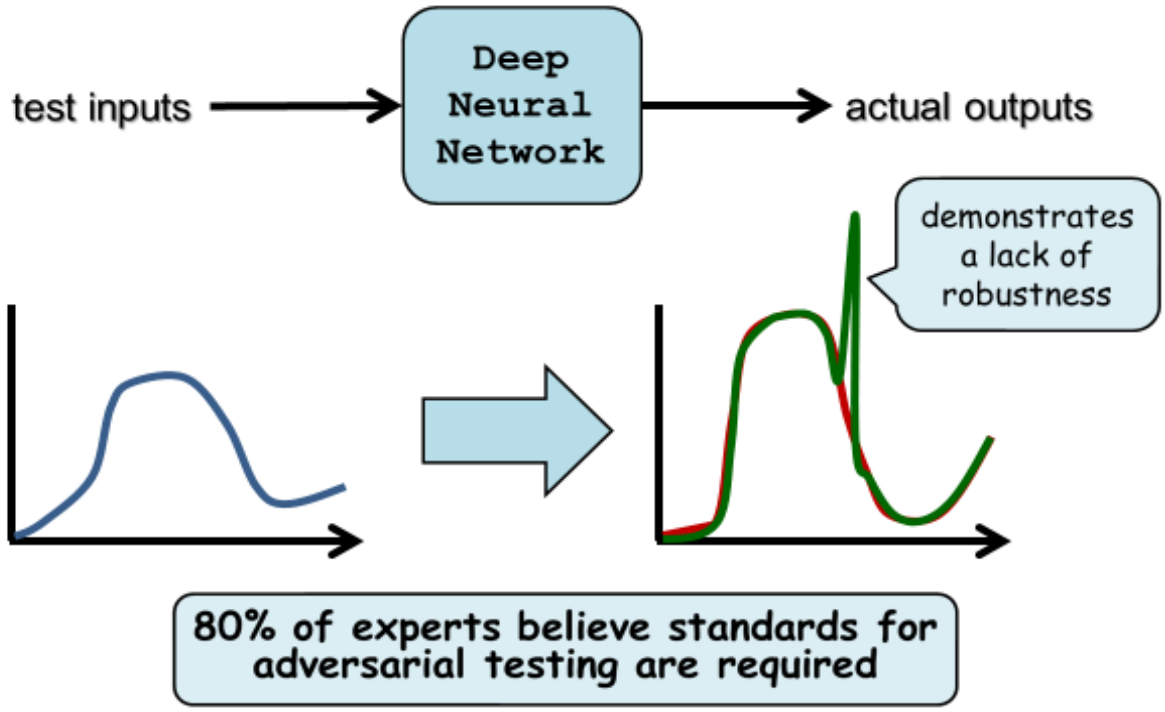
Ideal Deterministic Systems



Testing Non-Deterministic Systems

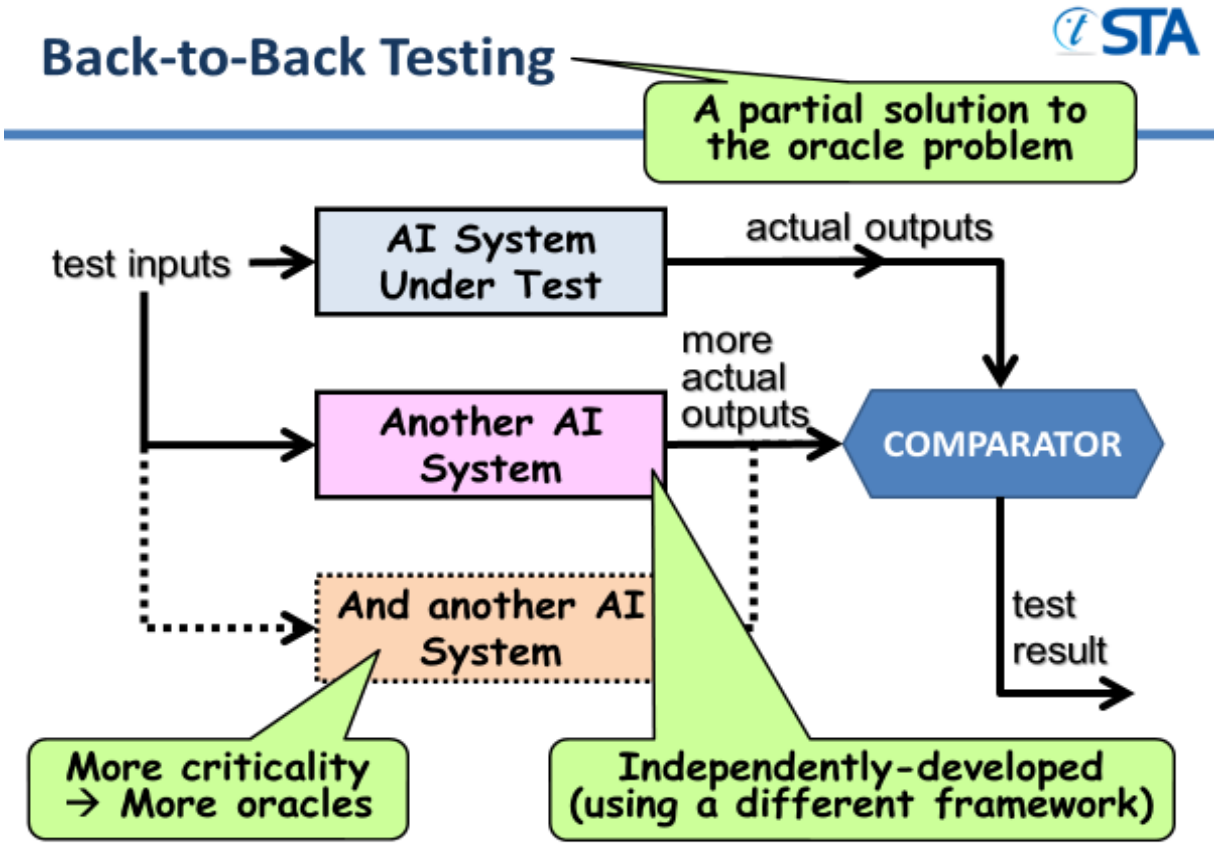


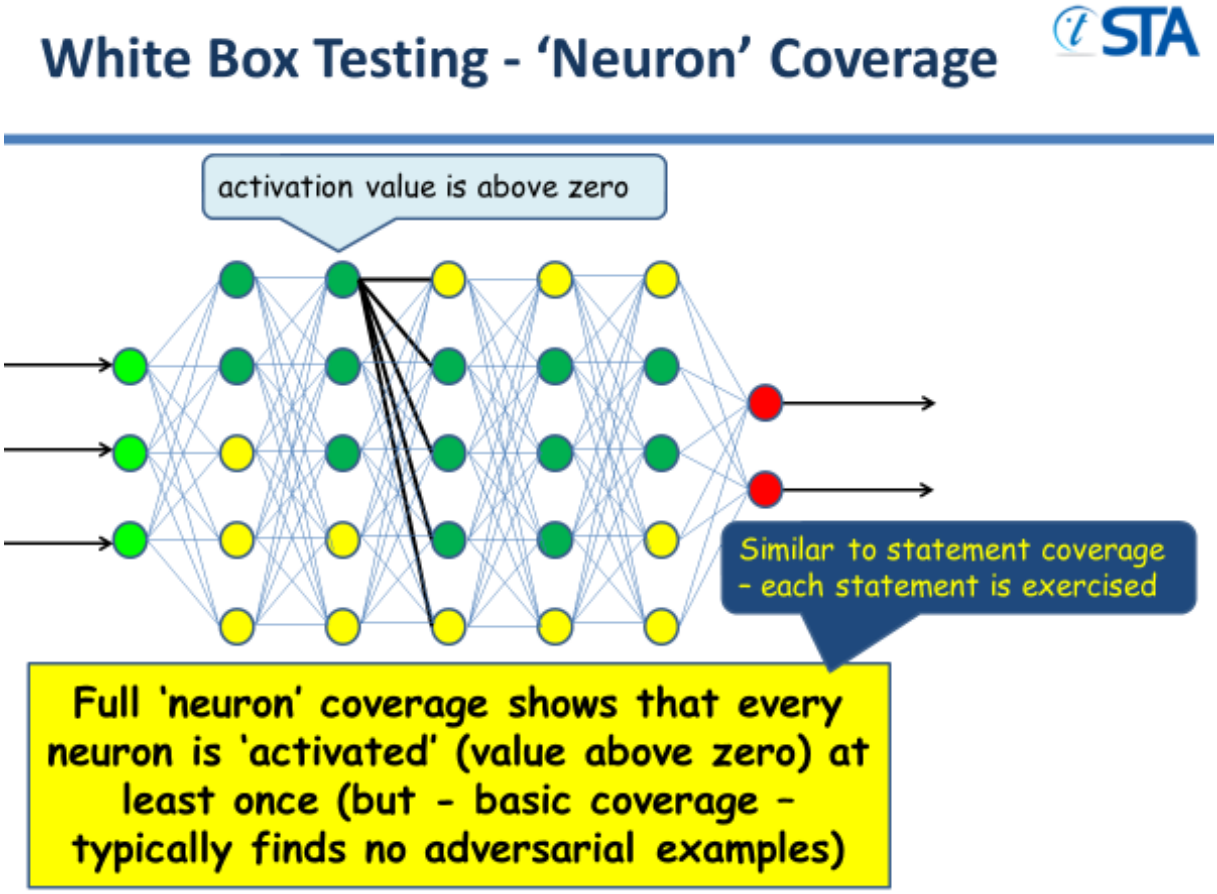
Adversarial Defects



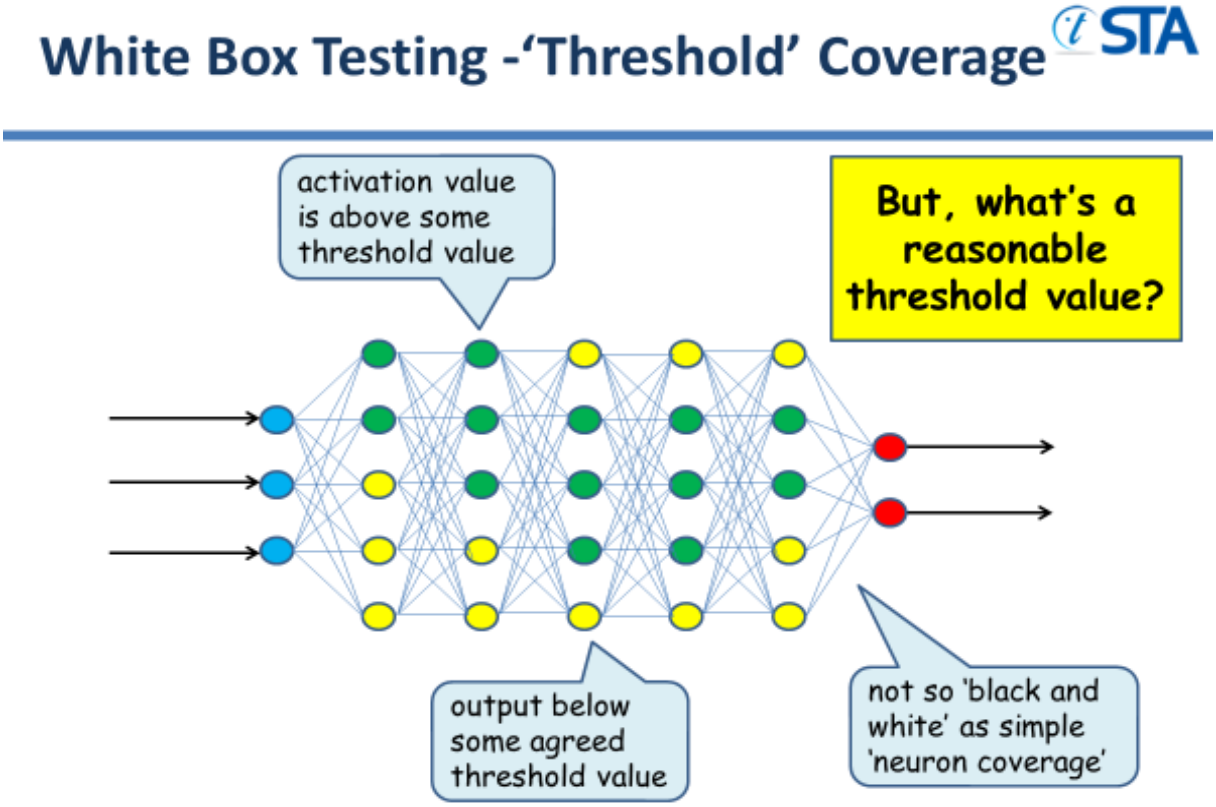
Black Box Test Challenges of AI Systems

- **Complexity and Test Oracle**
  - Often we cannot define required behaviour, e.g. how we…
    - recognize particular objects (e.g. child, paper bag, bike, obstruction)
    - 'know' that another car will move in a particular direction
    - plan a manoeuvre into moving traffic
    - know what to do in a new situation
- **Probabilistic Systems and Non-Determinism**
  - The probabilistic nature means that predicting expected results is difficult
  - Non-determinism causes real problems for regression testing
- **Autonomous Self-Learning Systems are difficult to test because they are difficult to understand (not just by testers!)…**

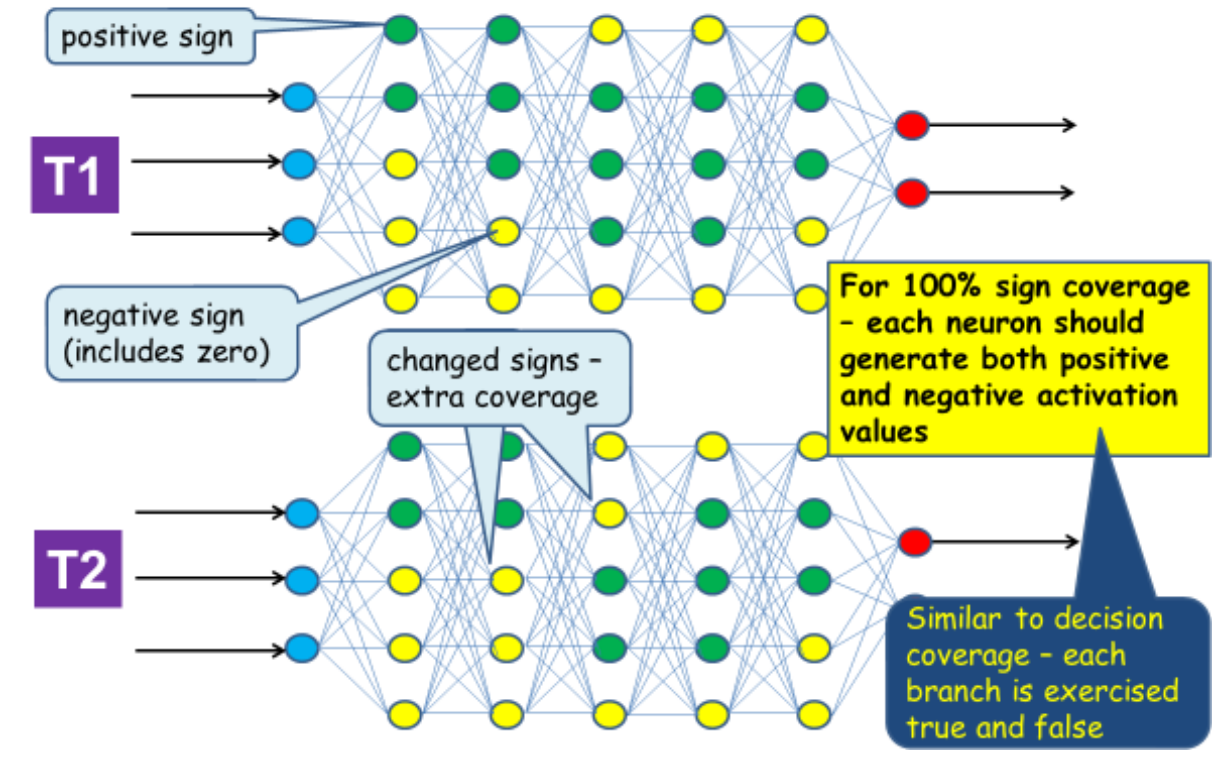

Back-to-Back Testing
A partial solution to the oracle problem
test inputs
AI System Under Test
actual outputs
more actual outputs
Another AI System
COMPARATOR
And another AI System
test result
More criticality → More oracles
Independently-developed (using a different framework)


White Box Testing - 'Neuron' Coverage
activation value is above zero
Similar to statement coverage - each statement is exercised
Full 'neuron' coverage shows that every neuron is 'activated' (value above zero) at least once (but - basic coverage - typically finds no adversarial examples)

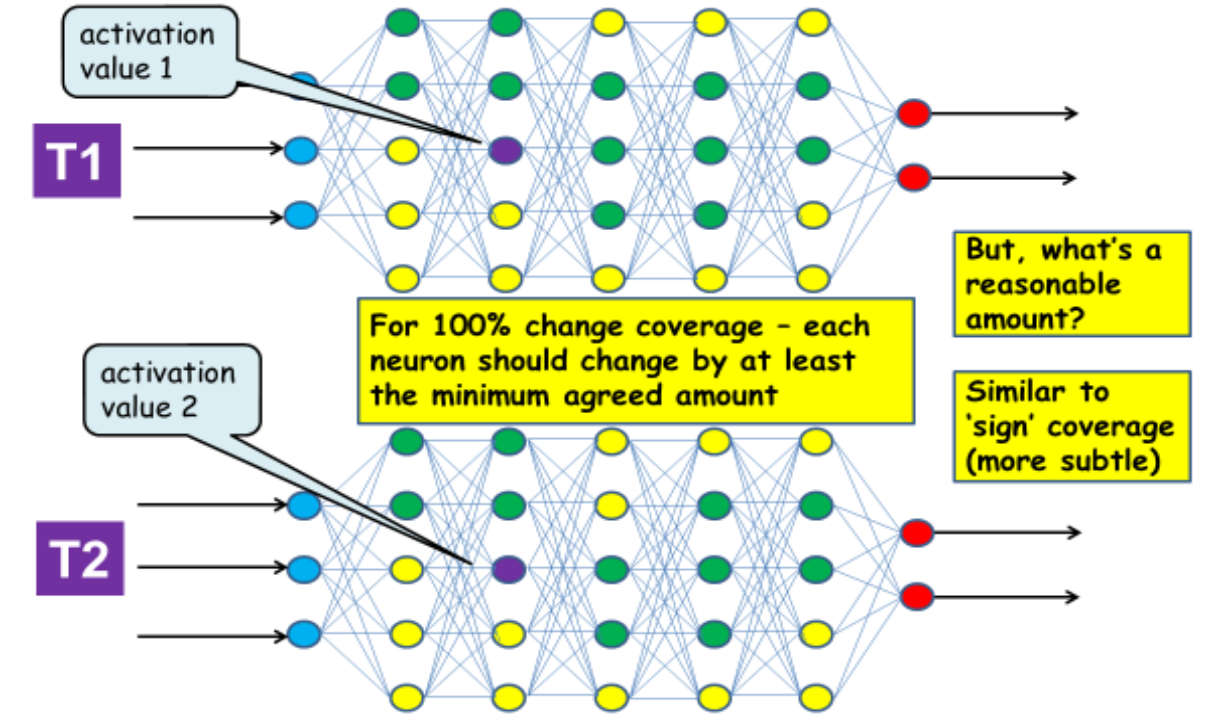
White Box Testing -'Threshold' Coverage



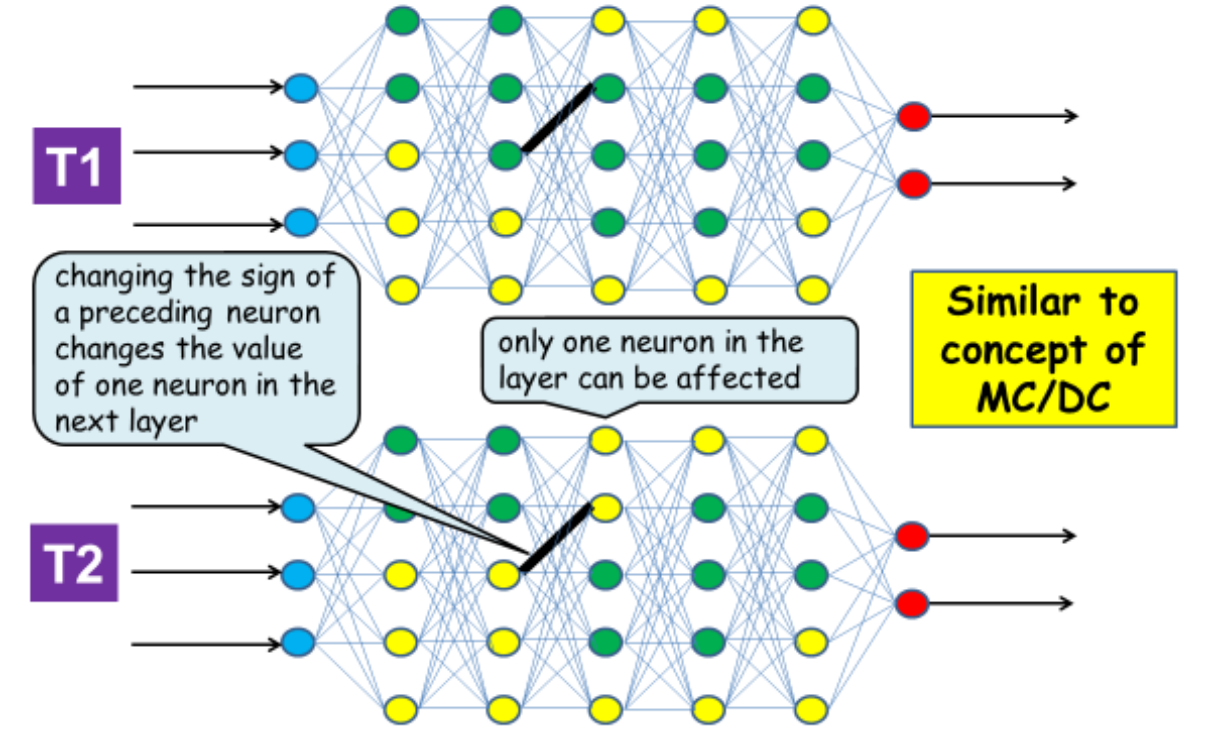
White Box Testing -'Sign' Coverage



White Box Testing -'Change' Coverage



White Box Testing – 'Sign-Sign' Coverage by Testing



Test Techniques for Neural Networks

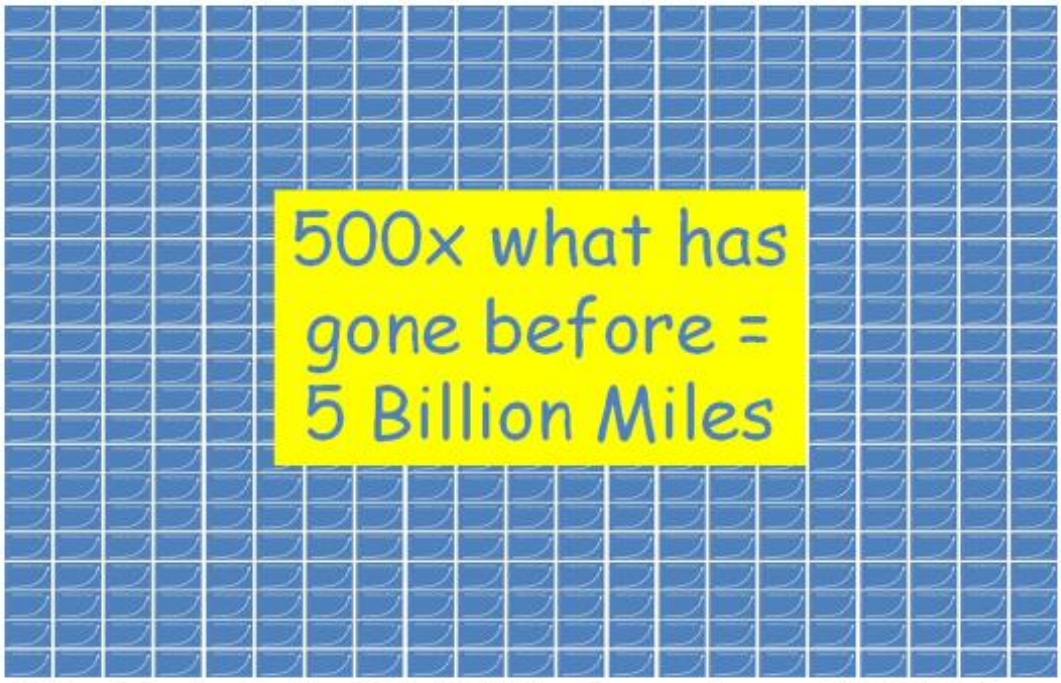
- **Random testing, traditional black box test techniques and neuron coverage do not appear to be good at finding adversarial examples**
  - but, similarly, random testing, equivalence partitioning and statement/branch coverage are also poor at finding defects in traditional embedded systems
- **But, adversarial examples are not our only problem**
- **Until we get more experience, we should use a mix of black and white box techniques**
  - with serious support for automation and test environments…

Applying AI Testing to Autonomous Cars

Combinatorial Testing Problems

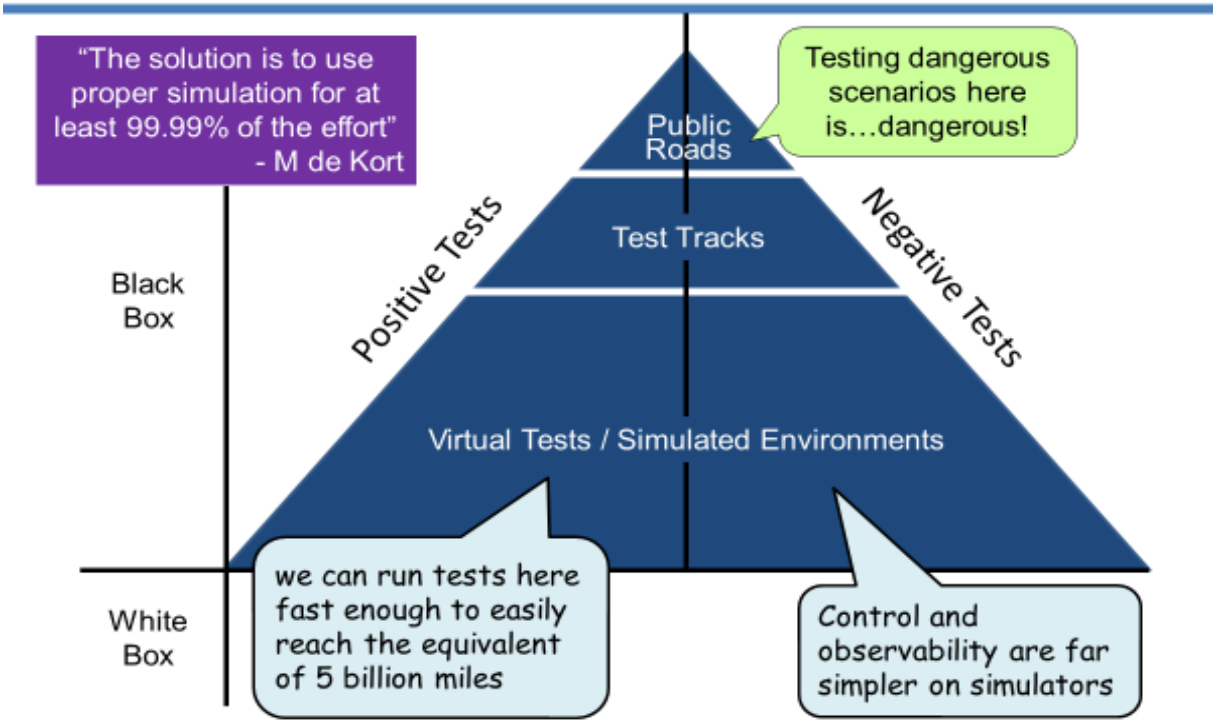
- **The number of inputs to an autonomous car is extremely high**
  - external sensor inputs (e.g. cameras, radar, lidar, etc.)
  - V2V communications / other external communications
  - internal information from the vehicle (e.g. engine)
  - map data
  - etc.
- **Even simple pairwise testing could generate millions of tests**

For AI to be 20% Better (than human drivers)



*Driving to Safety – How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability? RAND CORP. Nidihi Kalra, Susan M. Paddock

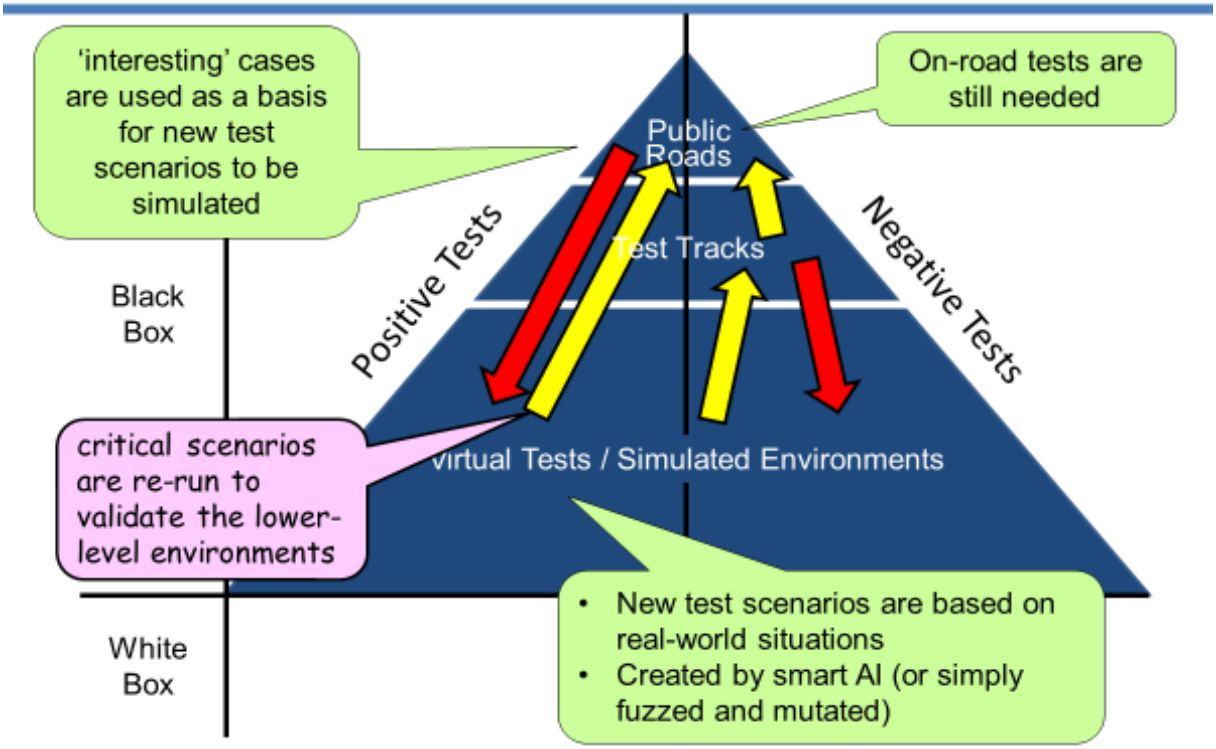
Autonomous Cars – Test Environments



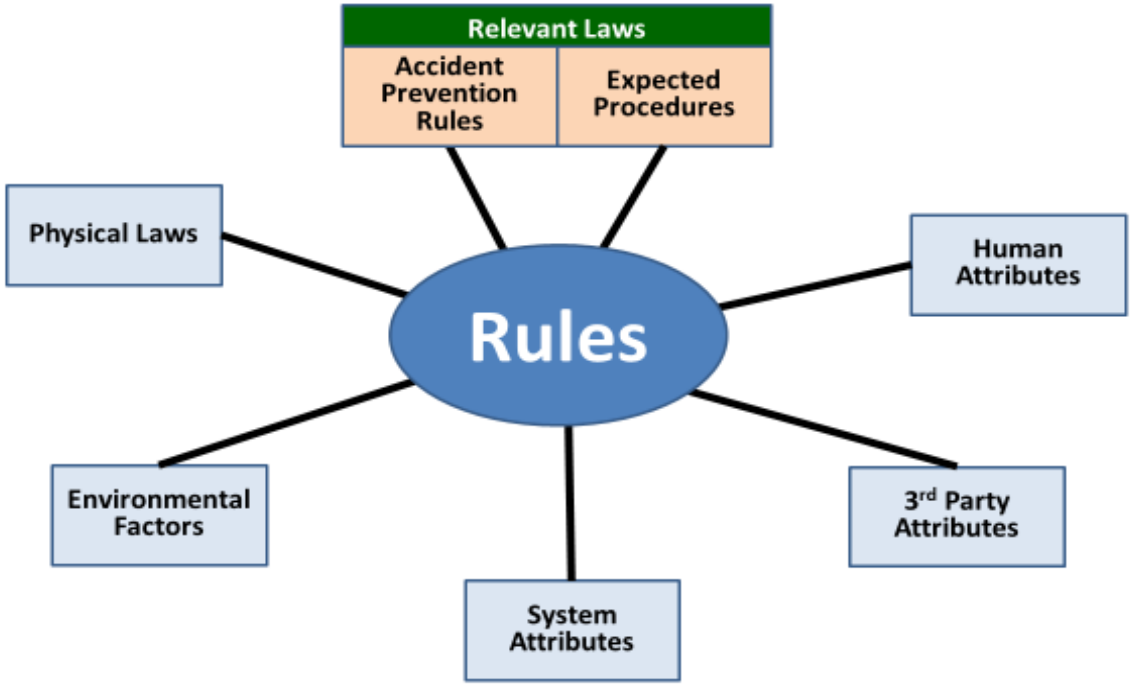
The Necessity of Virtual Test Environments

- **Pros**
  - Controllability
    - can easily generate the required test scenarios
  - Monitorability
    - anything can be monitored
  - Safety
    - no-one gets hurt and nothing gets broken
  - Speed
    - can run test scenarios far faster than real-time
- **Cons**
  - Cost of Fidelity
    - To make the simulator 'real' enough
  - Realism
    - Users know it is not real

Scenario Identification & Validation



Autonomous System Safety Rules



Context for Autonomous Functions

- **Defines each autonomous function's capability limits, e.g.**
  - Roadway types (interstate, local, etc.) on which the function is intended to operate safely;
  - Geographic area (city, mountain, desert, etc.);
  - Speed range;
  - Environmental conditions in which the function will operate (weather, daytime/night-time, etc.); etc.
- **Example**
  - US NHTSA 'Federal Automated Vehicles Policy', published in 2016, defines the 'Operational Design Domain (ODD)' for autonomous cars

Conclusions - Testing AI Systems

- **We need many tests to cope with these systems':**
  - probabilistic nature
  - high complexity
  - criticality (for safety-related use)
- **…and need the support of sophisticated virtual test environments**
- **Test techniques for Deep Neural Networks need much further research**
  - especially empirical studies of test effectiveness
  - but not concentrated on adversarial examples
- **Until we reach maturity, we should use a safety net…**

고품질 무기체계 소프트웨어를 위한 신뢰성 시험 환경 구축에 관한 연구

최민관 O, 박다운, 국승학
국방과학연구소
mkchoi@add.re.kr, dwpark90@add.re.kr, kuk@add.re.kr

A Study on Reliability Test Environment Setup for High Quality Weapon System Software

Minkwan Choi O, Daun Bak, Seunghak Kook
Agency for Defense Department

요 약

최근 무기체계에서 소프트웨어가 차지하는 비중이 증가됨에 따라 소프트웨어의 품질이 매우 중요한 요소가 되고 있다. 무기체계 소프트웨어의 품질 향상을 위해 무기체계 소프트웨어 개발 및 관리 매뉴얼에 소프트웨어 신뢰성을 제도화 하였고, 구체적인 방법 및 절차가 제시하고 있다. 이 매뉴얼에서 요구하는 신뢰성시험을 충족하기 위해서는 개발 전(全)순기에 걸쳐 지속적인 시험을 통해 결함의 검출 및 수정이 필요하지만, 보안을 위한 망분리 환경, 시험 도구 확보를 위한 비용 문제로 인해 적정 수준의 시험 환경을 구축하고 있지 못하다. 따라서 본 연구에서 한정된 환경 및 자원을 활용해 효율적으로 신뢰성시험을 수행 할 수 있는 환경 구축 방안을 제시하고자 한다.

1. 서 론

여느 산업과 유사하게 방위산업 내에서도 소프트웨어의 중요성이 날이 갈수록 증가하고 있다. 무기체계가 수행하고자 하는 대부분의 기능들이 소프트웨어를 기반으로 구현되고 있으며, 무기체계 연구개발 기간 중 상당수의 시간이 소프트웨어의 개발에 할애하고 있다. 또한 무기체계 연구개발의 근간이 되는 “무기체계 소프트웨어 개발 및 관리 매뉴얼” 역시 지속적으로 개선되고 있다. 해당 매뉴얼은 무기체계 소프트웨어 개발 관리를 위해 2014년 최초 제정되었으며, 무기체계 소프트웨어 개발 프로세스와 활동들에 대해 정의하고 있다[1].

특히 매뉴얼 개정의 핵심은 소프트웨어 품질과 신뢰성 확보를 위한 소프트웨어 신뢰성시험의 강화이다. 그러나 신뢰성시험이 강화됨에 따라 종전의 방식보다는 구현단계부터 시험을 지속적으로 수행하여 결함을 제거하는 것이 필요하다. 도구를 추가적으로 도입하여 개정된 매뉴얼에 대응할 수 있지만, 현실적으로 비용 문제가 있기 때문에 한정된 도구들을 적시에 활용할 수 있는 방안의 모색이 필요하다. 따라서 본 연구에서는 소프트웨어 신뢰성시험을 효율적으로 수행 할 수 있는 통합개발 시험 환경 구축에 관한 연구를 수행하고자 한다.

2. 소프트웨어 신뢰성시험 현황 및 문제점

무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 신뢰성시험을 크게 정적시험과 동적시험으로 구분하여 정의하고 있다. 정적시험이란 소프트웨어를 실행하지 않은 상태에서 잠재적인 결함을 검출하는 시험으로 코딩규칙, 취약점 점검, 소스코드 메트릭(Metric) 점검을 포함한다. 동적시험은 소프트웨어를 실제하드웨어에 탑재한 상태에서 시험절차서에 기술된 절차에 따라 소프트웨어 코드 실행률(Coverage)을 점검하는 것을 의미한다. 소프트웨어의 품질과 신뢰성 확보를 위해 신뢰성시험의 기준을 강화하는 방향으로 표 1과 같이 매뉴얼이 개정되었다[1-3].

표 1 무기체계 소프트웨어 신뢰성시험 범위의 변화

무기체계 소프트웨어 개발 및 관리 매뉴얼		’14.02 제정	’16.07 개정
정적 시험	코딩규칙	무기체계 SW 코딩 규칙 준수	국제표준(MISRA-C/C++ 등) 준수
	취약점 점검	CWE 658/659중 선별 적용	CWE 658/659/660 모두 준수
	소스코드 메트릭	없음	매뉴얼 상에 정의된 메트릭 종류 및 제한값 준수
동적 시험	대상	위험 분석 후 일정 수준 이상 위험 소프트웨어 항목 설정	결함의 발생빈도, 영향성, 제어가능성 평가 국제표준(DO-178등) 적용시 해당표준 준수
	목표값	위험 수준별 목표값 설정	결함의 빈도, 영향성, 제어가능성 평가표에 의한 코드 실행률 달성
	방법	구조기반시험 (Statement, Branch, MC/DC)	실제 하드웨어를 탑재한 상태에서 요구사항 기반으로 코드 실행률 점검

개정된 신뢰성시험 기준을 충족하기 위해서는 구현 초기단계부터 자동화 도구를 확보하여 지속적으로 시험을 수행해야 한다. 하지만 일반적으로 한정된 예산으로 인해 연구개발 기관들은 신뢰성시험을 위한 도구를 소량 확보하여 다른 사업과 공유하여 사용하고 있다. 이로 인해 개발자는 도구를 적시에 사용할 수 없을 뿐만 아니라, 소프트웨어 통합

시점에 시험을 수행하고 수정하는 방식을 취하고 있다. 그러나 매뉴얼에서 요구하는 신뢰성시험 기준을 만족하기 위해서는 개발 초기부터 시험을 수행해야 하며, 이를 위해서는 자동화 도구를 적시에 사용할 수 있는 환경 구축이 필요하다.

3. 소프트웨어 신뢰성시험 환경 구축 방안

소프트웨어 신뢰성시험 환경은 CI(Continuous Integration)를 이용하여 구축하고자 한다. CI는 소프트웨어 공학 관점에서 지속적인 품질 제어(Quality Control)를 적용하는 프로세스로써, 개발 중인 소스코드를 기존 소스코드와 통합 시 발생하는 문제를 조기에 식별하고 해결하기 위한 방법이다[4-6].

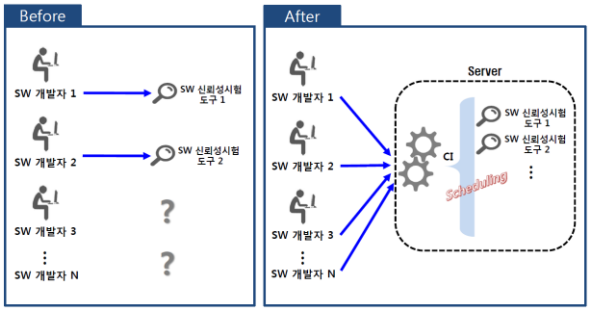


그림 1 SW신뢰성시험 환경 구축 전/후 비교

그림 1은 CI를 이용하여 신뢰성시험 환경 구축 전과 후를 비교한 것이다. 환경 구축 전에는 개발부서에서 신뢰성시험 도구를 보유하지 못하거나 소량의 도구만을 보유하고 있기 때문에 도구를 선점한 개발자가 시험을 모두 수행할 때까지 다른 개발자는 시험을 적시에 수행하지 못하는 문제가 있었다. 환경 구축 후에는 각각의 개발부서에서 보유한 도구들을 서버에 통합 구축하여 신뢰성시험이 필요한 개발자에게 우선순위에 따라 서비스를 제공함으로써 기존의 문제를 극복할 수 있다. 또한, CI에서 소스코드의 변경 유무를 지속적으로 확인하여, 빌드(Build) 및 신뢰성 시험을 수행하여 결과를 개발자에게 송신한다. 이와 같은 환경 구축을 통해 이해관계자(Stakeholder) 간 소스코드 및 시험 결과에 대한 모니터링과 추적성 관리가 가능할 것이라 기대된다.

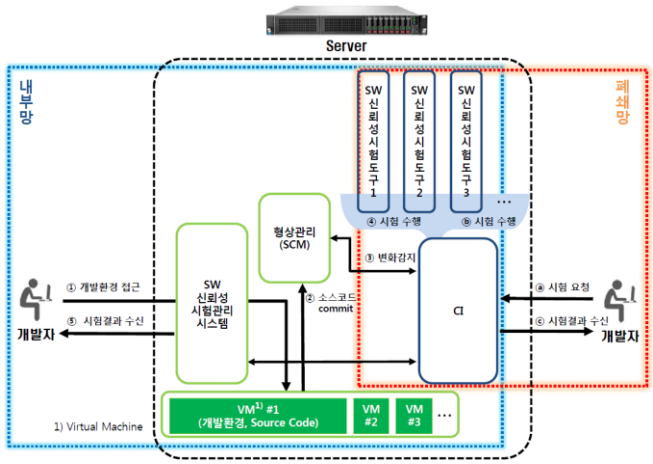


그림 2 소프트웨어 신뢰성시험 환경(안)

시험환경 구축에 앞서 무기체계 소프트웨어 개발 환경에 존재하는 제약사항들에 대한 이해가 필요하다. 우선 개발품목이 다양하고 많은 이해관계자가 사업에 참여하기 때문에 다양한 개발환경 및 테스트 도구에 대한 고려가 필요하다. 다음으로 보안에 대한 철저한 검토가 필요하다. 기본적으로 무기체계 연구개발은 인터넷망과 분리된 폐쇄망에서 이루어지고 있다. 이와 같은 제약사항을 충분히 고려하여 소프트웨어 신뢰성시험 환경 구축이 필요하다. 본 연구에서는 위의 사항들을 고려하여 그림 2와 같은 폐쇄망과 내부망을 지원하는 신뢰성시험 환경 구축 방안을 제안한다.

폐쇄망은 개발 PC와 도구가 설치된 서버를 물리적으로 연결한다. 별도의 형상관리나 개발환경 구축 없이 CI를 신뢰성시험 도구의 스케쥴링에 활용한다. 폐쇄망 환경은 서버에 개발환경 구축에 투입되는 시간과 비용을 절약할 수 있고, 한정된 수량의 도구를 효율적으로 사용 할 수 있다. 반면에 폐쇄망 환경 구성을 위해 테스트 장소가 서버가 있는 위치에 한정된다는 단점이 존재하며 형상관리(SCM, Software Configuration Management) 환경 부재로 개발된 소스코드와 시험결과에 대한 관리가 어려울 수 있다.

내부망은 서버 내 가상머신을 이용하여 각각의 개발환경을 구축하여 별도로 개발 PC의 연결 없이 신뢰성시험이 가능한 환경이다. 또한 소프트웨어 신뢰성시험 관리 시스템 기능을 서버에 구현하여 개발환경 및 소스코드의 형상 관리, 빌드/시험 결과에 대한 실시간 모니터링 기능을 제공하고자 한다. 내부망 환경에서는 사용자 집중에 따라 서버 성능 문제가 발생할 수 있지만, 환경 구축 이전보다 수월하게 신뢰성시험이 가능할 것으로 기대된다. 하나의 서버에서 폐쇄망과 내부망 사용자들에게 서비스를 제공하고, 서버를 통해 전사적으로 내부망에 서비스를 구축함에 따라 발생할 수 있는 보안 이슈를 대응하기 위해 보안 취약점과 관련된 추가적인 시험과 사용자 별 접근 권한에 대한 엄격한 관리가 필요하다.

4. 결 론

본 논문에서는 무기체계 개발에서 수행하는 소프트웨어 신뢰성시험을 효율적으로 수행 할 수 있는 시험 환경 구축 방안을 제안하였다. 망분리 환경에서 신뢰성시험 도구의 제한 없이 시험을 수행할 수 있는 환경을 구축함으로써 소프트웨어에 대한 지속적인 시험이 가능하며, 무기체계 소프트웨어의 품질과 생산성 향상을 기대할 수 있다.

참고문헌

[1] 방위사업청 매뉴얼 제2018-7호, 무기체계 소프트웨어 개발 및 관리 매뉴얼.
[2] MISRA-C/C++ [Online]. Available: https://www.misra.org.uk
[3] CWE 659/659/660 [Online]. Available: https://cwe.mitre.org
[4] CI [Online]. Available: https://ko.wikipedia.org/w/index.php?title=지속적_통합&oldid=20682038
[5] Jenkins [Online]. Available: https://jenkins.io
[6] Hudson [Online]. Available: http://Hudson-ci.org

ADD
AGENCY FOR DEFENSE DEVELOPMENT

KCSE 2019

무기체계 SW 신뢰성 통합관리시스템 소개

2019. 1.
공용기술센터 SW신뢰성기술실

국방과학연구소 Agency for Defense Development

CTC

Agenda

- 개요
- 무기체계 SW 신뢰성 통합관리시스템 소개
- 결론

국방과학연구소 Agency for Defense Development

CTC

RAM(Reliability, Availability, Maintainability)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ **RAM 정의**

✓ 시스템의 세가지 관련 특성과 이에 대한 운영 지원을 의미

✓ 신뢰성(Reliability), 가용성(Availability), 유지보수성(Maintainability)

(DOD, "DOD Guide for Achieving Reliability, Availability, and Maintainability", 2005)

▶ **RAM 업무 절차**

✓ 제품의 수명주기 전 단계에 걸쳐 수행

시스템 신뢰성 정의

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ **시스템 신뢰성**

✓ 특정 환경에서 특정 시간 동안 시스템이 고장(failure)에 자유로울 수 있는 확률

✓ 대표적인 시스템 신뢰성 표현 지표/단위

- 특정 시간 당 발생 가능한 고장률 (Failure Rate)
- 평균 수명 (MTTF, MTBF)

- MTTF : Mean Time To Failure
- MTTR : Mean Time To Repair
- MTBF : Mean Time Between Failure

✓ 시스템 고장(Failure)을 유발하는 결함(Defect)이 줄어들 수록 신뢰성은 증가

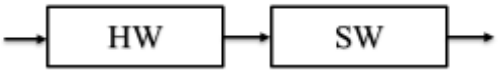
시스템 신뢰성 분석 : HW/SW 신뢰성 통합

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ Reliability Block Diagram

(IEEE, "IEEE 1633 – Recommended Practice for Software Reliability", 2016)

✓ **Reliability(System) = Reliability(HW) * Reliability(SW)**



- (예) R(Navigation Subsystem) = R(HW_GPS) * R(SW_GPS) * R(HW_Computer) * R(SW_GUICharting) * R(HW_DataStore) * R(SW_DataStore)

▶ 217plus Reliability Prediction Models

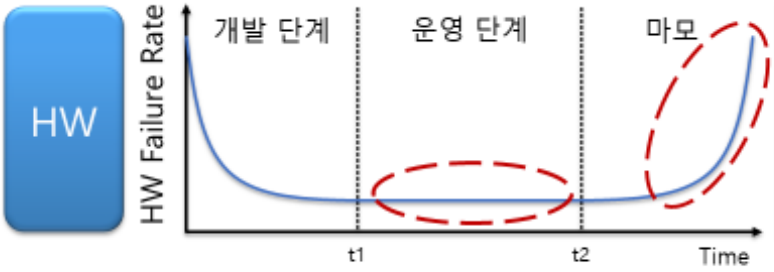
(RiAC, "Handbook of 217 Plus Reliability Prediction Models", 2006)

✓ **Reliability(System) = Reliability(HW) + Reliability(SW)**

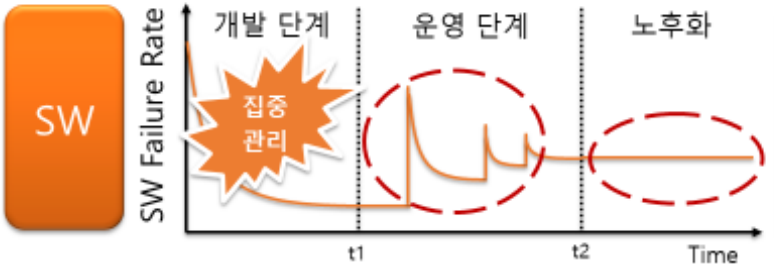
하드웨어 신뢰성 VS. 소프트웨어 신뢰성

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 하드웨어(HW) 신뢰성과 소프트웨어(SW) 신뢰성의 차이



- 배포(양산) 이후에는 수정이 쉽지 않아 개발자의 실수로 인한 결함 추가 가능성이 낮음
- 운영 시간이 지남에 따라 마모로 인한 결함이 발생



- 배포(양산) 이후에도 수정이 용이하여 안정화 기간 동안 개발자의 실수로 인한 결함의 추가 가능성이 높음
- 마모로 인한 결함은 발생하지 않음

✓ **서로 특징이 다른 HW 신뢰성과 SW 신뢰성을 별도 관리할 필요성 존재**

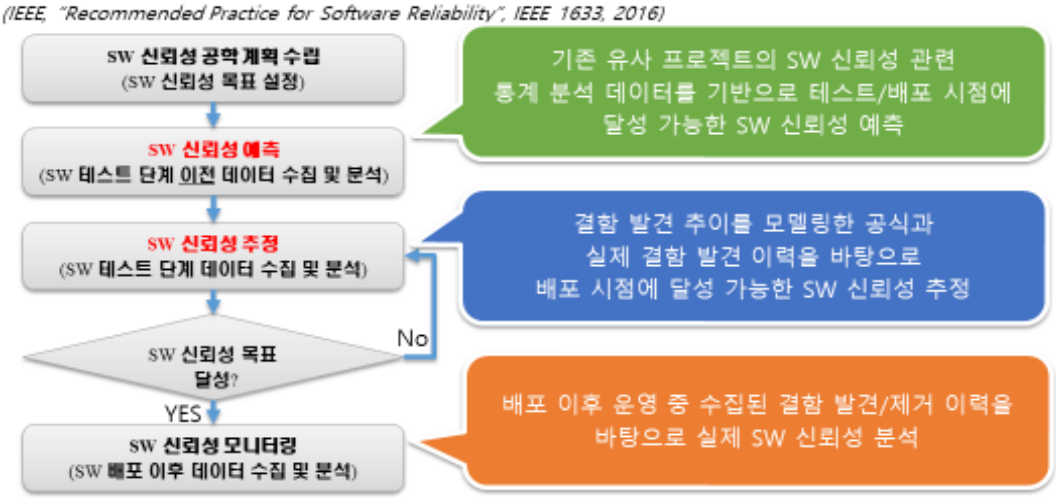
SW 신뢰성 공학 정의

ADD AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 공학

✓ **SW의 신뢰성을 구체화하고 예측하며 평가하기 위해 개발 및 운영 기간 동안 수집된 데이터에 통계적 기술을 적용하는 것**

(IEEE, "Recommended Practice for Software Reliability", IEEE 1633, 2016)



SW 신뢰성 예측 활동 (1/2)

ADD AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 예측 모델 (Software Reliability Prediction Model)

✓ **SW 테스트 단계 이전에 과거 유사 프로젝트의 통계 분석 데이터를 기반으로 현 프로젝트의 SW 결함 밀도를 예측하는 모델**

- Input: 개발 환경 정보와 관련된 설문지 작성
- Output: 테스트/운영 단계 진입 시 예측되는 결함 밀도(Defect Density)

✓ **예) RADC(Rome Air Development Center) Model**

- SW 개발 과정 중에 얻을 수 있는 다양한 정보에 대한 설문조사를 통해 현재 개발 중인 SW의 결함 밀도를 예측하는 모델

Input

아래 항목에 대한 설문지 작성
- SW Application Type 정보 **(A)**
- SW 개발 환경 정보 **(D)**
- SW 산출물의 특징 정보 **(S)**
  - 예외처리 능력 (SA)
  - 추적성 유지 여부 (ST)
  - 품질 검토 수행 결과 (SQ)
  - 개발 언어 유형 (SL)
  - SW 모듈화 정도 (SM)
  - SW 복잡도 (SX)
  - SW 개발 표준 준수 정도 (SR)

Model Logic

각 항목 별로 설문 결과를 수치화하여 모델의 결함 밀도 산출 식에 대입

Defect Density = A * D * S
(S = SA*ST*SQ*SL*SM*SX*SR)

Output

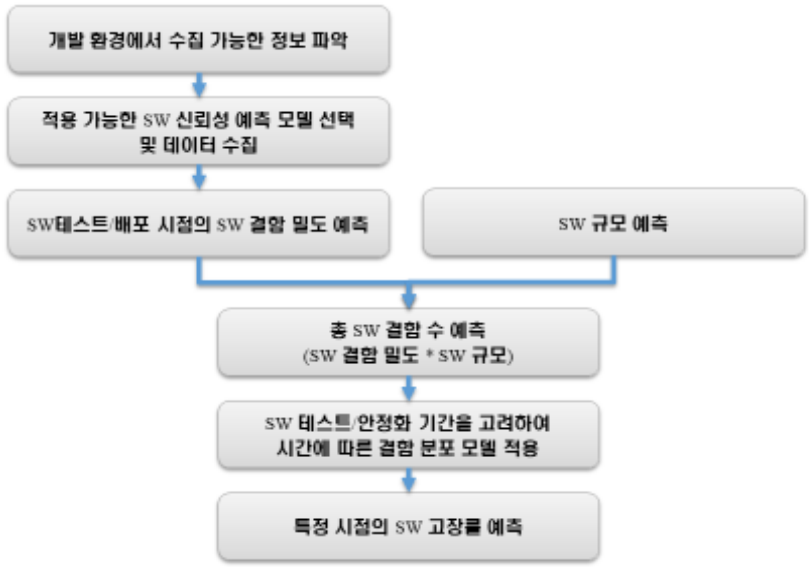
현재 개발 중인 SW의 예측 결함 밀도
Defect Density

※ 최종 SW 고장률 산출을 위해 SW 규모 예측 및 시간에 따른 결함 분포 모델 적용 등의 추가적인 활동 수행 필요

SW 신뢰성 예측 활동 (2/2)

AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 예측 모델을 이용한 SW 고장률 산출 절차



SW 신뢰성 추정 활동 (1/2)

AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 추정 모델 (Software Reliability Estimation Model 또는 Software Reliability Growth model, SRGM)

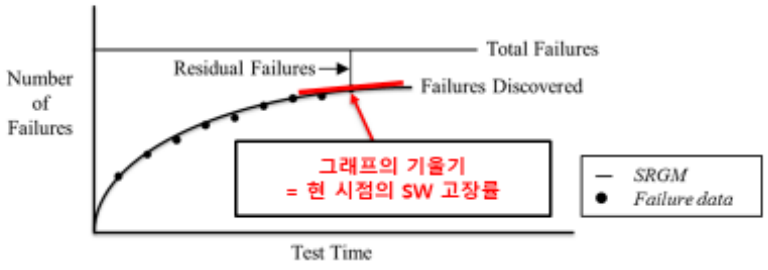
✓ 결함 발견 추이를 모델링한 공식에 SW 테스트/운영 단계에서 실제 수집되는 결함 발견 이력 데이터를 대입하여 현 프로젝트의 SW 고장률을 추정하는 모델

- Input : 테스트 회 차 당 테스트 수행 소요 시간 및 발견된 결함 수
- Output: 추정되는 잔존 결함 수, 현 시점의 SW 고장률 등

SRGM
Mean value function
Ex) $m(t) = a(1-e^{-bt})$

결함 데이터

테스트 시간	결함 수
1주	4
2주	9
3주	12
…	…



SW 신뢰성 추정 활동 (2/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ **SW 신뢰성 추정 모델을 이용한 SW 고장률 산출 절차**

✓ 다수의 모델을 적용하여 적합성(Goodness of Fit) 검정을 한 후 최적의 모델을 선택하여 결과 산출

SW 테스트/운영 단계에서 발견되는 결함 이력 데이터 수집

↓

적용 가능한 다수의 SW 신뢰성 추정 모델 선정

↓

결함 이력 데이터를 모델에 대입하여 각 모델의 매개변수 추정 (Model Fitting)

↓

적합성 검정을 통한 최적의 모델 선택

↓

현재 혹은 특정 시점의 SW 고장률 추정

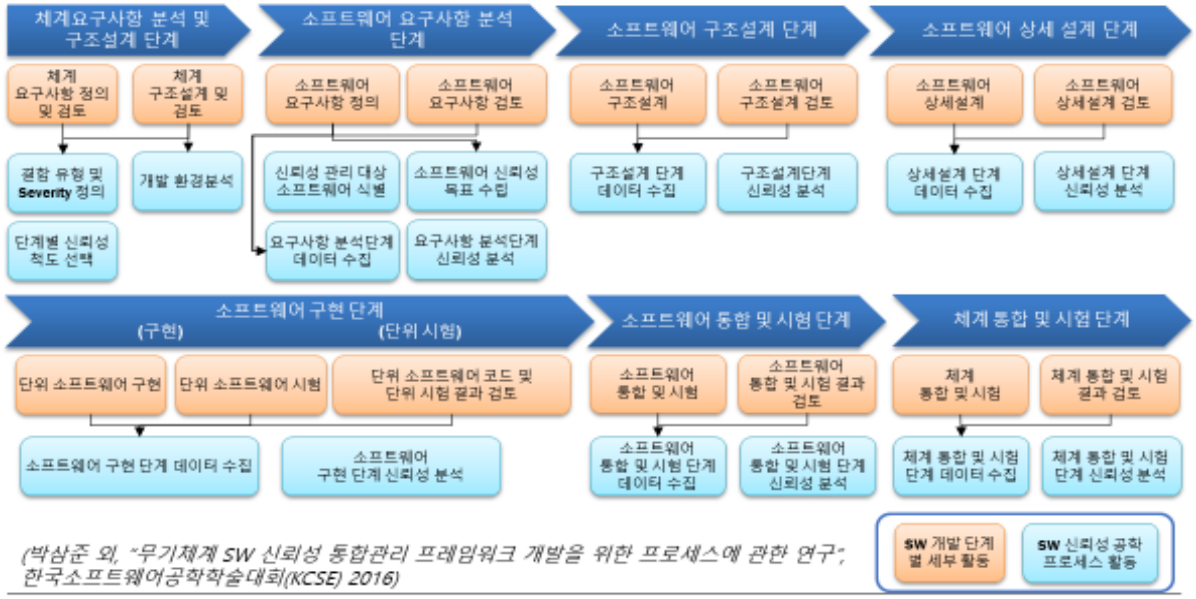
연구 접근 방법

ADD
AGENCY FOR DEFENSE DEVELOPMENT

무기체계 SW 신뢰성 공학 프로세스 정립

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SW 개발 단계 별 주요 SW 신뢰성 공학 활동



(박삼준 외, "무기체계 SW 신뢰성 통합관리 프레임워크 개발을 위한 프로세스에 관한 연구", 한국소프트웨어공학학술대회(KCSE) 2016)

무기체계 SW 신뢰성 공학 척도 정의 (1/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SW 신뢰성 척도 목록 및 설명

✓ SW 신뢰성과 직/간접적으로 연관이 있는 척도 28종 선정

(김태현 외, "무기체계 소프트웨어 신뢰성 향상을 위한 척도에 관한 연구", 군사과학기술학회 종합학술대회, 2017)

척도 번호	척도 명	설 명
SM01	Defects by Type	SW 개발 단계 별 수집되는 결함의 유형별 분포
SM02	Defects by Severity	SW 개발 단계 별 수집되는 결함의 심각도 별 분포
SM03	**Predicted Testing Defect Density**	**SW 신뢰성 예측 모델을 통해 계산되는 테스트 시점의 예측 결함 밀도**
SM04	**Defect Density**	**현 단계에서 수집되는 실측 결함 밀도 (SW 규모 대비 결함 비율)**
SM05	Inspection Intensity	SW 규모 대비 Inspection 수행 시간의 비율
SM06	Requirements Compliance	SW 개발 단계 별 활동 수행을 통해 발생하는 결함 중 SW 요구사항과 관련된 결함의 유형별 비율
SM07	Requirements Traceability	현 단계에서 반영된 상위 요구사항(SW 혹은 시스템)의 비율 (SW 구현단계까지)
SM08	Requirements Change Rate	SW 개발 단계 별 활동 수행을 통해 변경된 SW 요구사항의 비율
SM09	Phase Containment Effectiveness for SW Requirement Analysis Phase	현재까지 발견된 SW 요구사항 관련 결함 중 SW 요구사항 분석 단계에서 발견된 결함의 비율
SM10	Cyclomatic Complexity	소스코드 내 함수 별 순환 복잡도

* **붉은색 굵씨** : SW 신뢰성 분석 모델을 통해 산출되는 척도
* **굵은 글씨** : SW 신뢰성과 직접적으로 연관 있는 척도

무기체계 SW 신뢰성 공학 척도 정의 (2/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SW 신뢰성 척도 목록 및 설명

✓ SW 신뢰성과 직/간접적으로 연관이 있는 척도 28종 선정

(김태현 외, "무기체계 소프트웨어 신뢰성 향상을 위한 척도에 관한 연구", 군사과학기술학회 종합학술대회, 2017)

척도 번호	척도 명	설 명
SM11	**Predicted Total Testing Defects**	**SW 신뢰성 예측 모델을 통해 계산되는 테스트 시점의 예측 총 결함 수**
SM12	Class Size	소스코드 내 클래스 별 규모
SM13	Method Inheritance Factor	소스코드 내 클래스 별 상속의 영향도
SM14	Depth of the Inheritance Tree	소스코드 내 클래스 별 상속의 깊이
SM15	Coupling Factor	소스코드 내 클래스 간 연결 정도
SM16	Code Coverage	소스코드의 테스트 커버리지 비율
SM17	**Fault Density**	**현 단계에서 수집되는 실측 결점 밀도 (SW 규모 대비 결점 비율)** **(※ Defect와 Fault를 구분하여 관리할 경우 적용)**
SM18	Modified Phase Screening Effectiveness for SW Requirement Analysis Phase	현재까지 발견된 SW 요구사항 관련 결함 중 SW 요구사항 분석 단계 이후 단계에서 발견된 결함의 비율 (각 개발 단계 별)
SM19	Defect Rate due to Bad Fix	현 단계에서 발견된 결함 중 Bad fix로 인해 발생한 결함의 비율
SM20	Cumulative Faults By Status	결함의 관리 상태(수정 중, 수정 완료 등) 별 분포

* **붉은색 글씨** : SW 신뢰성 분석 모델을 통해 산출되는 척도
* **굵은 글씨** : SW 신뢰성과 직접적으로 연관 있는 척도

무기체계 SW 신뢰성 공학 척도 정의 (3/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SW 신뢰성 척도 목록 및 설명

✓ SW 신뢰성과 직/간접적으로 연관이 있는 척도 28종 선정

(김태현 외, "무기체계 소프트웨어 신뢰성 향상을 위한 척도에 관한 연구", 군사과학기술학회 종합학술대회, 2017)

척도 번호	척도 명	설 명
SM21	Cumulative Test Cases Executed and Passed(for test i in each baseline)	전체 테스트 케이스 대비 실행 및 통과된 테스트 케이스 비율
SM22	**Estimated Remaining Defects**	**SW 신뢰성 추정 모델을 통해 추정되는 현 시점 기준 잔존 결함 수**
SM23	**Estimated Remaining Test Time to Achieve a Specified Failure Rate**	**SW 신뢰성 추정 모델을 통해 추정되는 목표 고장률 달성 시 까지 요구되는 테스트 시간**
SM24	**Mean Time To Repair**	**평균 수리 시간**
SM25	**Estimated Current Failure Rate**	**SW 신뢰성 추정 모델을 통해 추정되는 현 시점 기준 고장률**
SM26	Defects by Severity(for test i in each baseline)	SW 개발 단계 별 수집되는 결함의 유형별 분포 (세부 시간 단위 기준)
SM27	Requirements Coverage	현 단계에서 테스트 수행을 통해 검증된 SW 요구사항의 비율 (SW 구현단계 이후)
SM28	**Predicted/Estimated System Failure Rate***	**HW/SW 통합 신뢰성 분석 모델을 통해 추정되는 시스템 고장률**

* **붉은색 글씨** : SW 신뢰성 분석 모델을 통해 산출되는 척도
* **굵은 글씨** : SW 신뢰성과 직접적으로 연관 있는 척도

무기체계 SW 신뢰성 공학 척도 정의 (4/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 척도 별 적용 가능한 SW 개발 단계 정립

(김태현 외, "무기체계 소프트웨어 신뢰성 향상을 위한 척도에 관한 연구", 군사과학기술학회 종합학술대회, 2017)

척도 번호	척도 명	SW 개발 단계					
		요구사항 분석	구조설계	상세설계	구현	SW 통합 및 시험	체계 통합 및 시험
SM01	Defects by Type	○	○	○	○	○	○
SM02	Defects by Severity	○	○	○	○	○	○
SM03	**Predicted Testing Defect Density**	○	○	○	○		
SM04	**Defect Density**	○	○	○	○	○	○
SM05	Inspection Intensity	○	○	○	○	○	○
SM06	Requirements Compliance	○	○	○	○	○	○
SM07	Requirements Traceability	○	○	○	○		
SM08	Requirements Change Rate		○	○	○	○	○
SM09	Phase Containment Effectiveness for SW Requirement Analysis Phase	○	○	○	○	○	○
SM10	Cyclomatic Complexity				○		

* **붉은색 글씨 :** SW 신뢰성 분석 모델을 통해 산출되는 척도
* **굵은 글씨 :** SW 신뢰성과 직접적으로 연관 있는 척도

무기체계 SW 신뢰성 공학 척도 정의 (5/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 척도 별 적용 가능한 SW 개발 단계 정립

(김태현 외, "무기체계 소프트웨어 신뢰성 향상을 위한 척도에 관한 연구", 군사과학기술학회 종합학술대회, 2017)

척도 번호	척도 명	SW 개발 단계					
		요구사항 분석	구조설계	상세설계	구현	SW 통합 및 시험	체계 통합 및 시험
SM11	**Predicted Total Testing Defects**	○	○	○	○		
SM12	Class Size			○	○		
SM13	Method Inheritance Factor			○	○		
SM14	Depth of the Inheritance Tree			○	○		
SM15	Coupling Factor				○		
SM16	Code Coverage				○		
SM17	**Fault Density**				○	○	○
SM18	Modified Phase Screening Effectiveness for SW Requirement Analysis Phase			○	○	○	○
SM19	Defect Rate due to Bad Fix				○	○	○

* **붉은색 글씨 :** SW 신뢰성 분석 모델을 통해 산출되는 척도
* **굵은 글씨 :** SW 신뢰성과 직접적으로 연관 있는 척도

무기체계 SW 신뢰성 공학 척도 정의 (6/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 척도 별 적용 가능한 SW 개발 단계 정립

(김태현 외, "무기체계 소프트웨어 신뢰성 향상을 위한 척도에 관한 연구", 군사과학기술학회 종합학술대회, 2017)

척도 번호	척도 명	SW 개발 단계					
		요구사항 분석	구조설계	상세설계	구현	SW 통합 및 시험	체계 통합 및 시험
SM20	Cumulative Faults By Status				○	○	○
SM21	Cumulative Test Cases Executed and Passed(for test i in each baseline)				○	○	○
SM22	**Estimated Remaining Defects**					○	○
SM23	**Estimated Remaining Test Time to Achieve a Specified Failure Rate**					○	○
SM24	**Mean Time To Repair**				○	○	○
SM25	**Estimated Current Failure Rate**					○	○
SM26	Defects by Severity(for test i in each baseline)					○	○
SM27	Requirements Coverage					○	○
SM28	**Predicted/Estimated System Failure Rate**	○	○	○	○	○	○

* **붉은색 글씨** : SW 신뢰성 분석 모델을 통해 산출되는 척도
* **굵은 글씨** : SW 신뢰성과 직접적으로 연관 있는 척도

국방과학연구소 Agency for Defense Development

CTC - 19 -

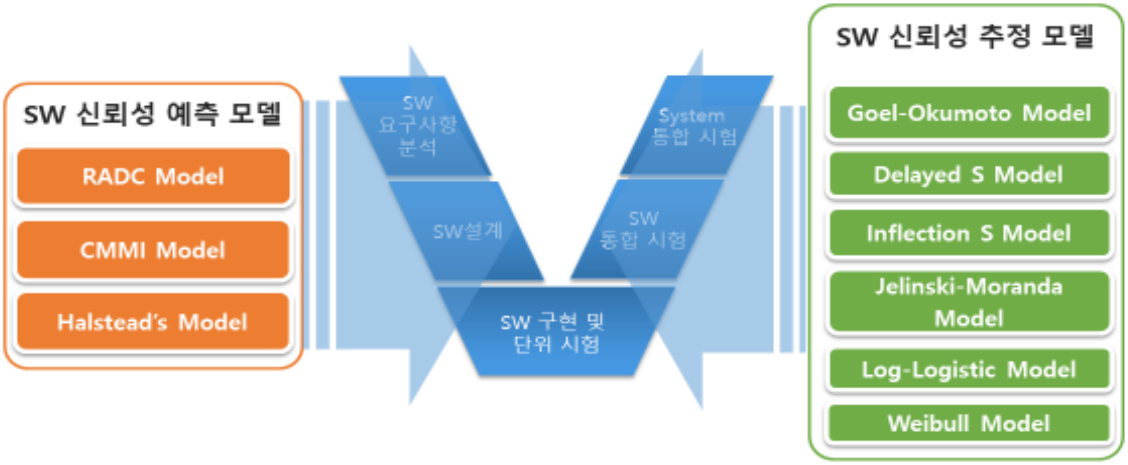
무기체계 SW 신뢰성 공학 모델 선정 (1/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ IEEE 1633에서 제시하는 SW 신뢰성 예측/추정 모델 선정

✓ SW 신뢰성 예측 모델 3종 기능 제공

✓ SW 신뢰성 추정 모델 6종 기능 제공



국방과학연구소 Agency for Defense Development

CTC - 20 -

무기체계 SW 신뢰성 공학 모델 선정 (2/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 예측/추정 모델 관련 척도 개념 및 흐름 정의

SW 신뢰성 예측 모델 관련 척도 (SW 테스트 단계 이전)

- 개발 환경에서 수집 가능한 정보 파악
- 적용할 SW 신뢰성 예측 모델 선택 및 데이터 수집
- **SM03** Predicted Testing Defect Density 산출
- SW 규모 데이터 수집
- **SM11** Predicted Total Testing Defects 산출

- 시간에 따른 결함 분포 모델 선택 및 데이터 수집
- SW 신뢰도 (Failure Rate) 산출
- HW 신뢰도 (Failure Rate) 수집
- **SM28** Predicted System Failure Rate 산출

SW 신뢰성 추정 모델 관련 척도 (SW 테스트 단계 이후)

- SW 테스트 과정 중 발견되는 결함 이력 데이터 수집
- 다수의 SW 신뢰성 추정 모델에 대한 매개변수 추정
- 적합성 검정을 통한 최적의 추정 모델 선택
- **SM22** Estimated Remaining Defects 산출
- **SM25** Estimated Current Failure Rate 산출

- 목표 SW 고장률 입력
- **SM23** Estimated Remaining Test Time to Achieve a Specified Failure Rate 산출
- HW 신뢰도 (Failure Rate) 수집
- **SM28** Predicted System Failure Rate 산출

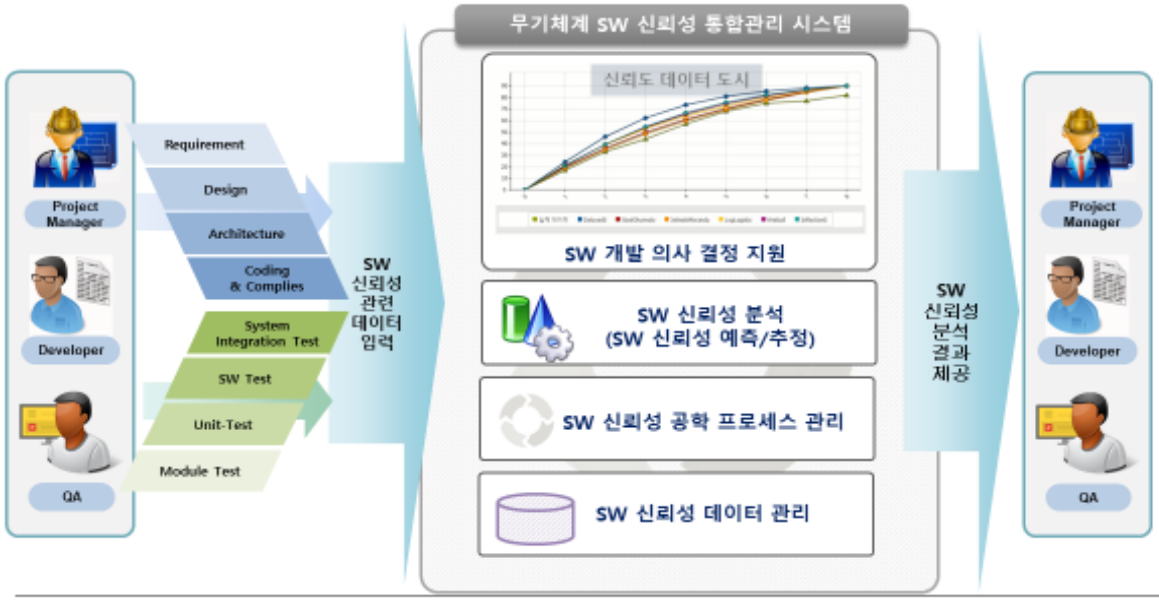
국방과학연구소 Agency for Defense Development

CTC - 21 -

무기체계 SW 신뢰성 통합관리시스템 개발

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 개발 전(全) 단계에서 신뢰성 공학 활동을 지원하기 위한 시스템 개발



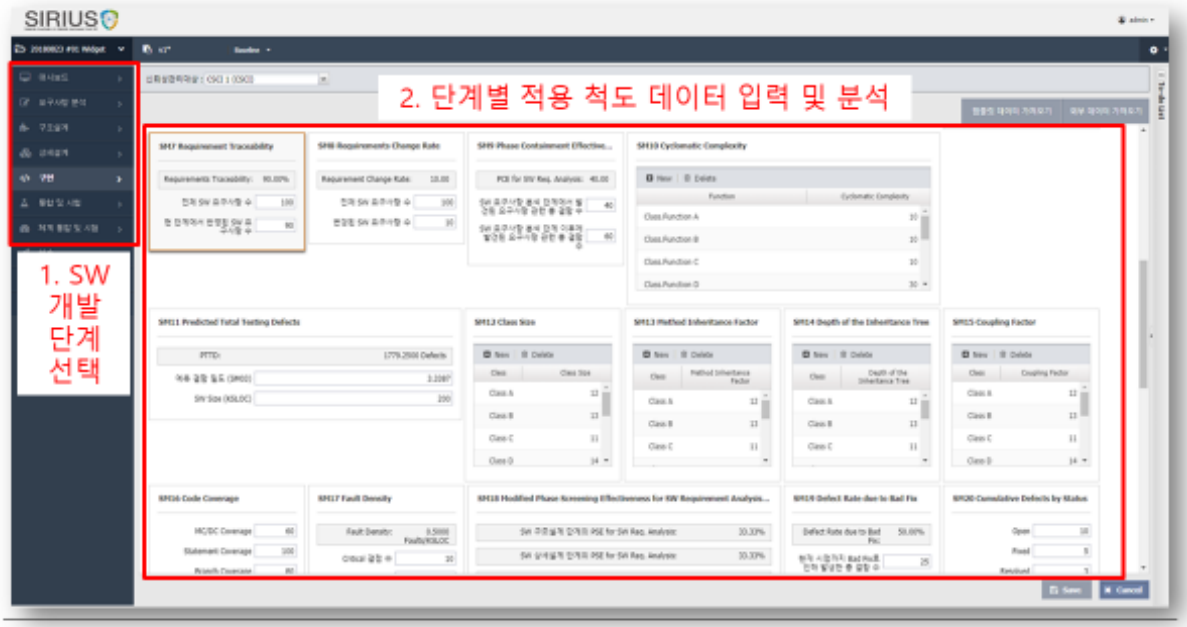
국방과학연구소 Agency for Defense Development

CTC - 22 -

SW 개발 전(全) 단계 데이터 관리 (1/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 개발 단계별 척도 데이터 입력 및 분석 기능 제공

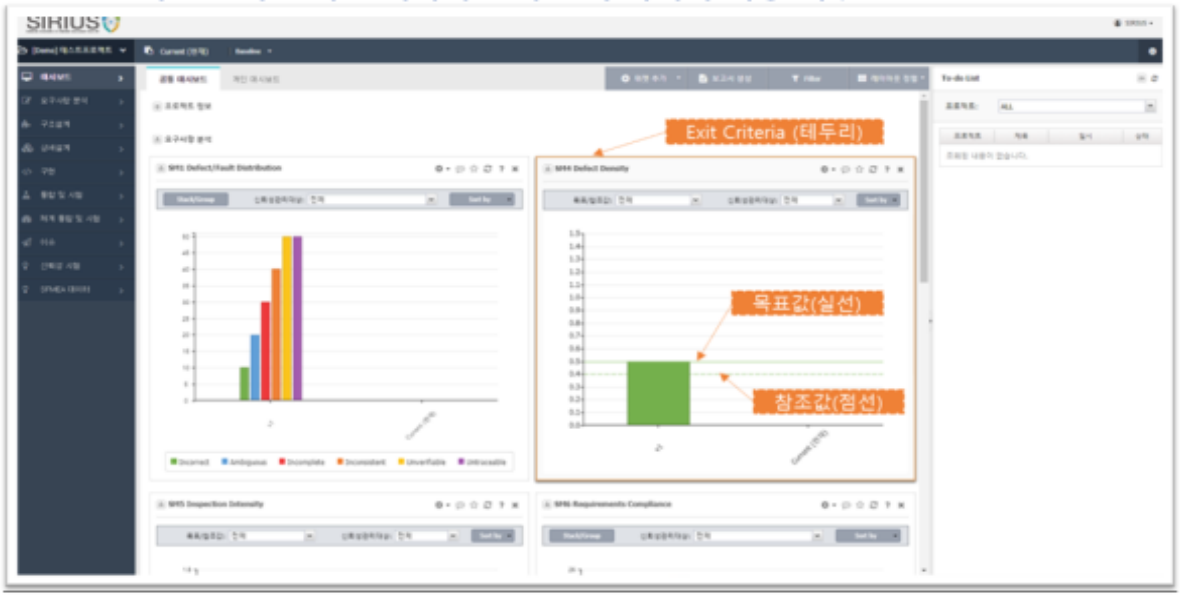


국방과학연구소 Agency for Defense Development

SW 개발 전(全) 단계 데이터 관리 (2/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 개발 단계별 척도 데이터 분석 결과 가시화 기능 제공

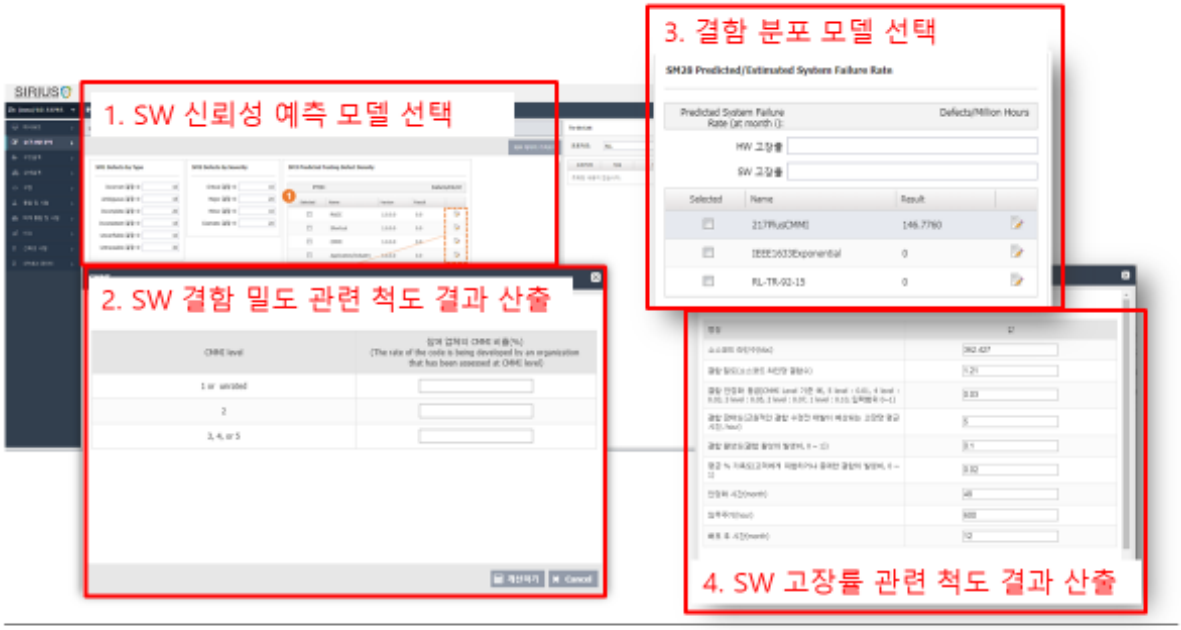


국방과학연구소 Agency for Defense Development

SW 신뢰성 분석 모델 지원 (1/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 예측 모델 관련 척도 분석 기능 제공

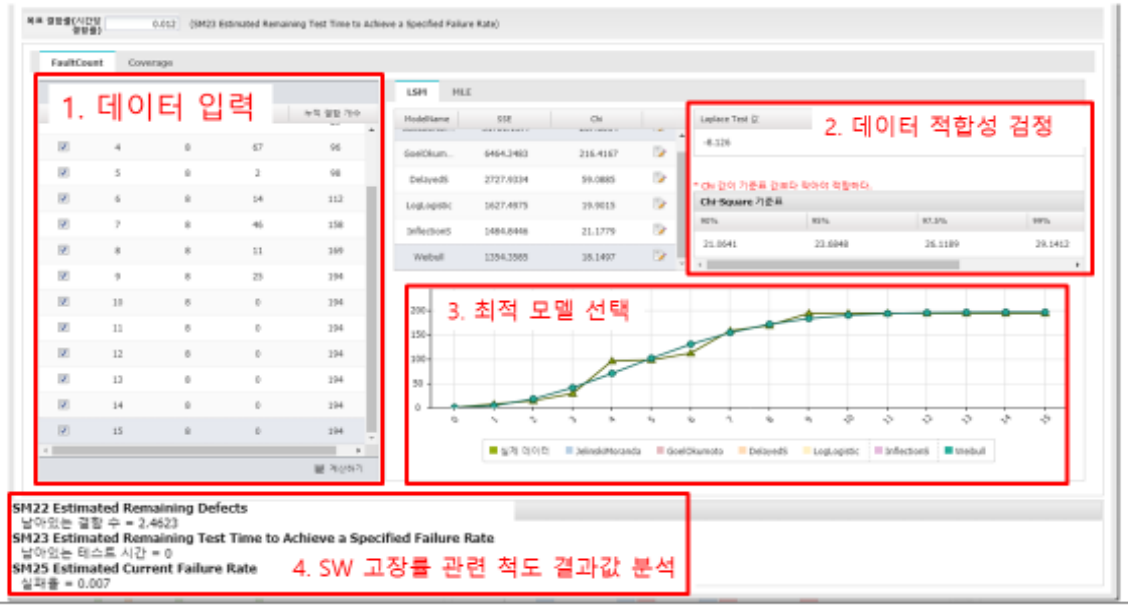


국방과학연구소 Agency for Defense Development

SW 신뢰성 분석 모델 지원 (2/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 신뢰성 추정 모델 관련 척도 분석 기능 제공



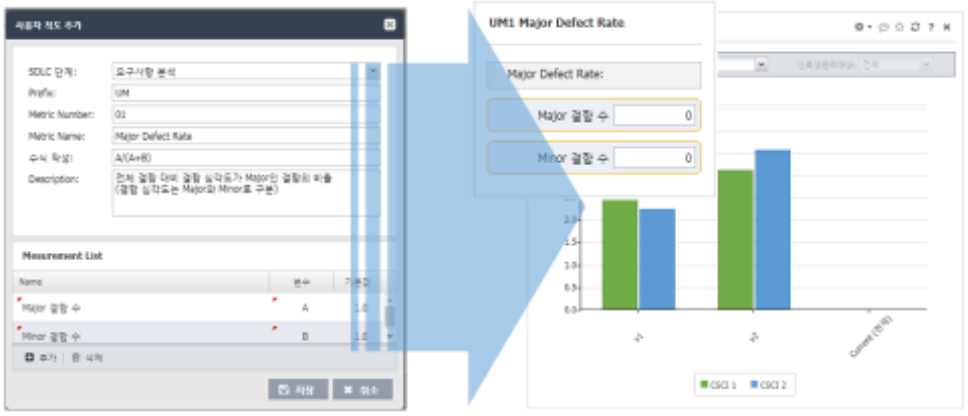
국방과학연구소 Agency for Defense Development

시스템 확장 : 척도 및 모델 추가

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 사용자 정의 척도 및 신규 모델 추가 기능 제공

✓ 단순 계산 척도 추가 기능 제공



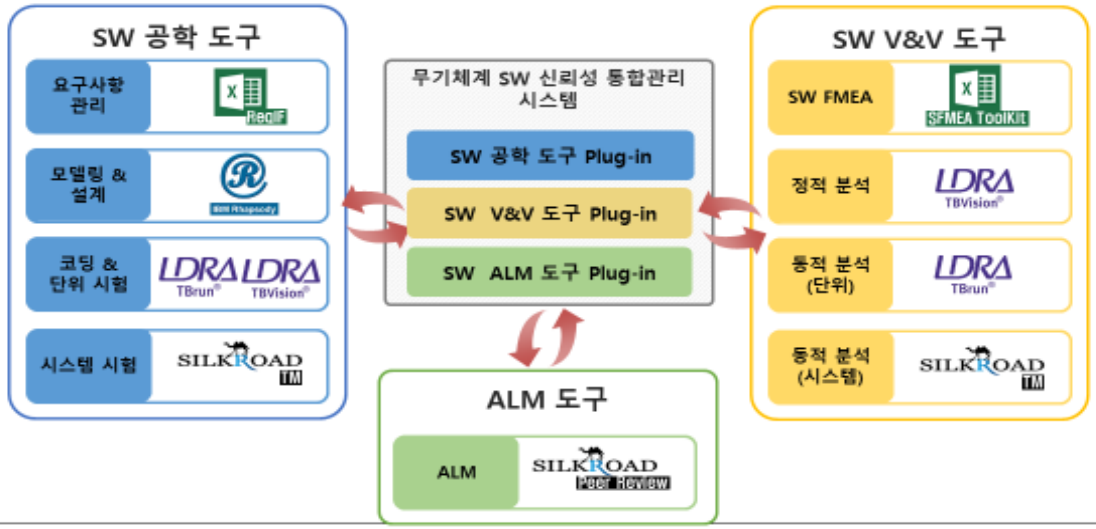
✓ 신규 모델 추가를 위한 표준 인터페이스 및 라이브러리 개발 가이드라인 제공

시스템 확장 : 외부 도구와의 연계 (1/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SW 공학 도구 5종 및 V&V 도구 4종, ALM 도구 1종 연동 기능 제공

✓ 외부 도구로부터 획득 가능한 데이터 자동 수집 기능 제공



시스템 확장 : 외부 도구와의 연계 (2/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 외부 도구 연동을 위한 표준 인터페이스 개발 및 Plug-in 가이드라인 제공

✓ 외부 도구 연동을 위한 Open API 18 종 제공

✓ 외부 도구 plug-in 추가 개발을 위한 가이드라인 제공

타 도구와의 비교

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 개발 전(全) 단계에 걸친 SW 신뢰성 공학 활동 지원 시스템 최초 개발

✓ 최신 SW 신뢰성 공학 관련 국제 표준을 적용하여 시스템 개발

도구 명	SW 개발 전(全) 단계 데이터 관리	SW 신뢰성 예측 모델 지원	SW 신뢰성 추정 모델 지원	HW/SW 통합 신뢰성 분석 모델 지원	외부 도구 데이터 연동 지원
본 과제 개발 시스템	O	O	O	O	O
Softrel, LLC	△	O	O	O	X
ReliaSoft	△	X	O	O	X
CASRE (Computer Aided Software Reliability Estimation)	△	X	O	X	X
GERT (Good Enough Reliability Tool)	△	X	X	X	O
SREPT (Software Reliability Estimation and Prediction Tool)	△	O	O	O	X

데모 영상

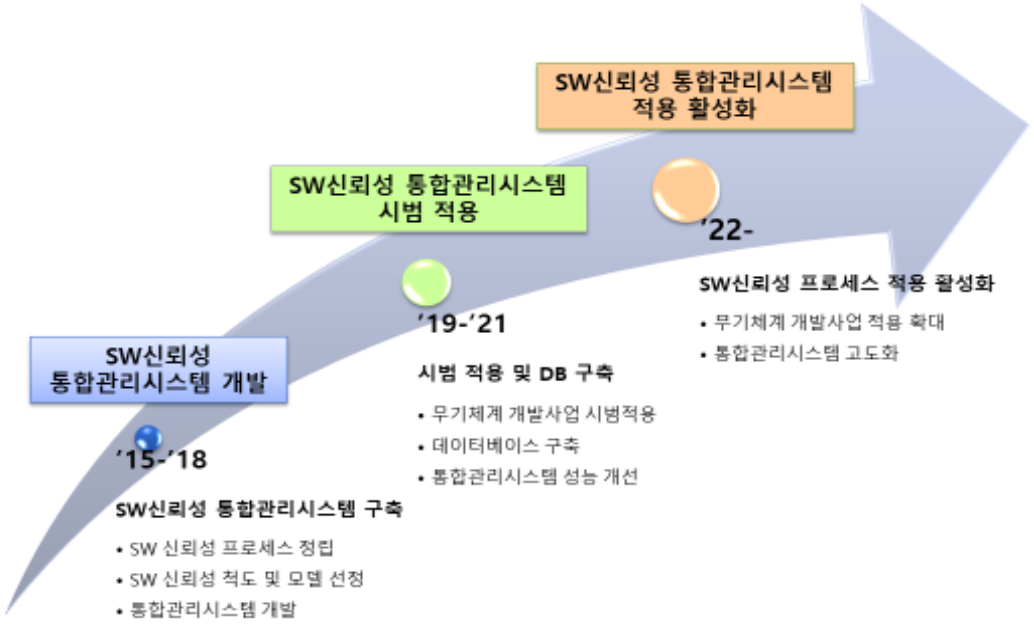
ADD
AGENCY FOR DEFENSE DEVELOPMENT

국방과학연구소 Agency for Defense Development

CTC - 31 -

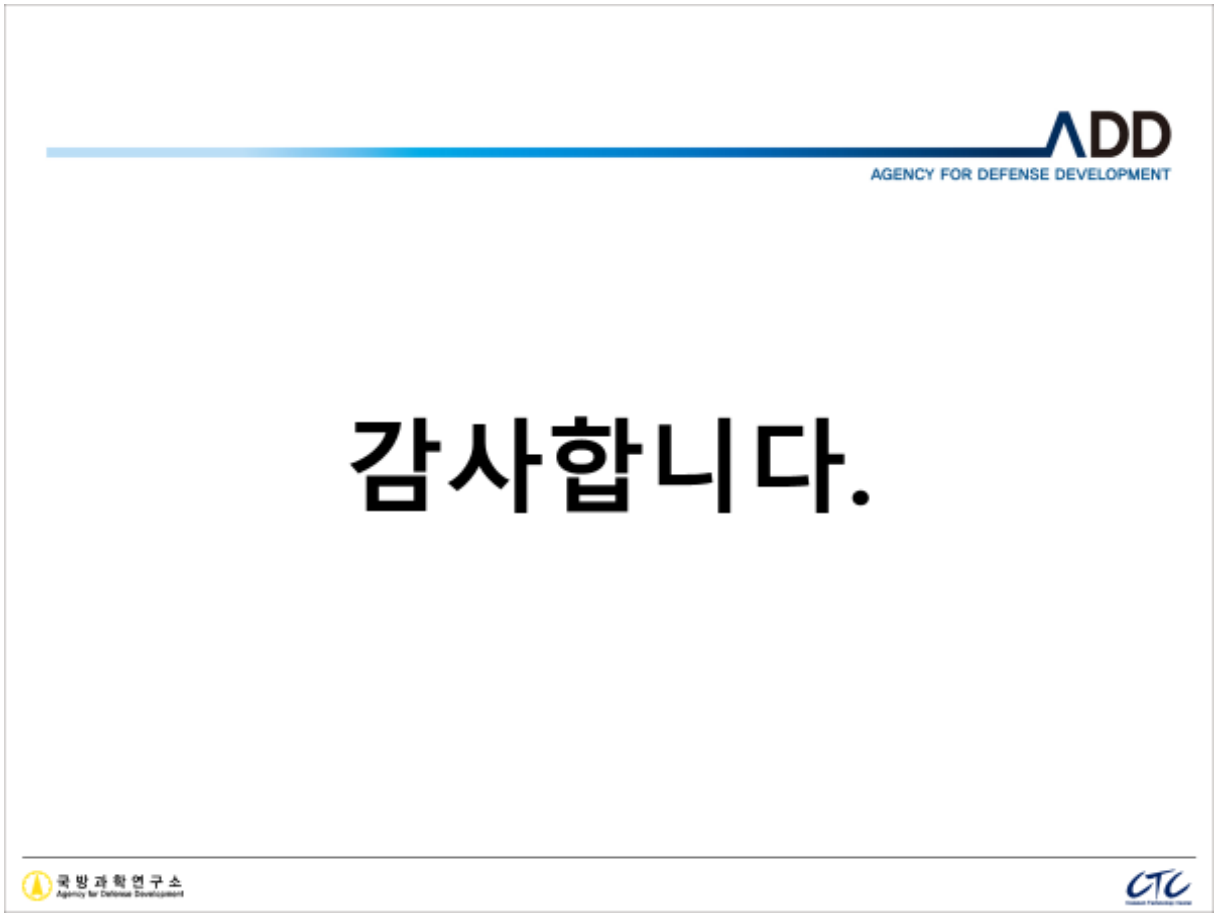
결론 및 발전 방향

ADD
AGENCY FOR DEFENSE DEVELOPMENT



국방과학연구소 Agency for Defense Development

CTC - 32 -


ADD
AGENCY FOR DEFENSE DEVELOPMENT
감사합니다.
국방과학연구소
Agency for Defense Development
CTC

고신뢰성 내장형 무기체계 소프트웨어를 위한 실행 가능한 모델링 언어

배정호 ○ 구봉주

국방과학연구소

jhbae@add.re.kr, tgkoo@add.re.kr

An Executable Modeling Language for a High Reliable Embedded Weapon Software

JungHo Bae○ BongJoo Koo

Agency for Defense Development

소프트웨어 개발 과정 중 구현 단계는 기존의 설계된 문서를 이해하여 컴퓨터를 위한 프로그래밍 언어를 사용하여 코드를 작성하는 단계로 많은 오류가 만들어 질 수 있다. 이를 방지하기 위하여 UML과 같은 정형화된 모델 기반의 설계 언어를 사용하여 설계하고, 정적, 동적 테스트를 통하여 코드에 숨어있는 오류를 발견하기 위한 노력을 하고 있다. 하지만 기존의 모델링 언어는 분석과 설계를 위하여 사용되기 때문에 실제로 실행하기에는 구체성이 낮으며, 구현 단계로 진입하면 유지보수가 힘들다는 한계가 있다. 그리고 코드 테스트는 프로그래밍 언어 고유의 특성에 대한 오류에 집중하고 있으며 요구사항을 명확히 구현하였는지 확인하기가 어렵다. 본 연구에서는 구현 단계의 문제점과 요구사항과 설계 검증의 한계를 극복하기 위하여 실행 가능한 모델링 언어를 제안한다. 제안한 모델링 언어의 효용성을 확인하기 위하여 제안한 언어를 실행하는 실행 엔진을 QT 프레임워크 기반으로 구현하였으며, 실제 해군지휘통제체계와 연동하는 프로그램을 제작하여 실제로 동작함을 확인하였다.

1. 서 론

무기체계 소프트웨어 개발은 요구사항분석, 구조설계, 상세설계, 구현, 통합 및 시험의 단계를 거친다 [1]. 요구사항분석, 구조설계, 상세설계의 결과물은 사람이 이해하는 것이 목적이므로 문서의 형태가 된다. 자연어 기반의 기술 문서는 모호하고 예상하지 못한 기능들의 충돌이나 누락된 기능들이 있을 수 있다. 더욱 명확하고 체계적인 기술을 위하여 UML과 같은 모델링 언어가 사용되기도 한다 [2].

구현 단계에서는 이전 단계의 산출물인 설계 문서를 이해하여 C++, Java와 같은 언어로 프로그램을 작성한다. 구현단계에서는 크게 두가지의 오류가 발생한다. 첫번째는 설계를 명확하게 이해하지 못해서 생기는 오류이다. 더 자세히 살펴보면, 기존의 설계 자체에 오류가 있을 수 있고, 설계에 반영된 내용을 누락하거나, 설계에 없는 내용이 추가되거나, 설계의 내용 자체를 틀리게 해석했을 수도 있다.

두번째는 프로그래밍 언어를 사용함으로 해서 생기는 오류이다. 프로그래밍 언어는 자연어와 다른 방식의 언어이기 때문에 많은 실수를 유발할 수 있다. 예를 들어 if(a = b) 와 if(a == b)는 전혀 다른 의미임에도 잘못 사용하는 실수를 하고, C++의 포인터와 같이 각 언어 고유의 특성에 따라 다양한 버그가 발생하기도 한다. 구현 단계의 오류를 줄이기 위하여 Rhapsody [3] 등의 도구를 이용하여 모델로부터 일부 코드를 자동으로 생성하는 기능을 활용하기도 한다.

방위사업청은 구현된 코드를 검증하기 위하여 무기체계 소프트웨어 개발 및 관리 매뉴얼을 통해 정적, 동적 테스트를 수행하도록 하고 있다. 정적 분석을 통해 코딩 규칙, 실시간 오류, 메트릭들을 확인하여 프로그래밍 언어 고유의 특성에 의한 오류를 줄이고, 동적 테스트를 통해 모든 코드를 최소 한 번 이상 실행함으로써 소프트웨어의 신뢰성을 높이는 것이 목적이다. 테스트의 신뢰성과 효율성을 높이기 위하여 QA C/C++ [4], CodeSonar [5], VectorCast [6], LDRA [7], Sparrow QCE [8], CodeScroll Inspector/Sniper/Tester [9] 등의 다양한 테스트 자동화 도구를 사용한다.

기존의 코드 기반 정적, 동적 테스트의 한계는 요구사항이나 설계를 검증하기가 어렵다는 것이다. 프로그램을 검증하기 위하여 코드 기반의 테스트를 진행하 듯 설계를 검증하기 위해서는 설계 기반의 테스트 케이스가 도출되어야 한다.

요구사항 기반의 테스트를 위하여 블랙박스 테스트를 수행하지만 체계적인 테스트 케이스 생성에는 한계가 있다. 기존의 설계는 자연어로 기술되어 있거나, UML 등의 모델링 언어를 사용하더라도 분석 및 설계 단계에서 작성된 모델은 의미있는 테스트 케이스가 생성될 정도로 세부적이지 않을 수 있기 때문이다. 뿐만

아니라 개발 과정 중 기능의 추가/수정 등이 발생하고 개발 기한에 대한 압박 등에 의하여 설계와 구현 간의 차이가 발생하기도 한다.

따라서 본 연구에서는 구현 단계의 문제점과 설계 검증에 대한 문제점을 극복하기 위한 방법으로 실행 가능한 모델링 언어를 제안한다. 제안하는 모델링 언어를 사용함으로써 프로그램밍 언어 사용에 대한 부담과 이를 검증하기 위한 노력을 줄이고, 설계에 집중함으로써 소프트웨어의 신뢰도를 높이는 것이 목적이다.

본 논문의 구성은 다음과 같다. 2장에서는 실행가능한 모델링 언어를 정의하고 예제를 소개한다. 3장에서는 해당 언어의 실효성을 확인하기 위하여 구현한 실행 엔진을 설명하고, 4장에서 결론을 논의한다.

2. 실행가능한 모델링 언어

본 논문에서 제안하는 모델링 언어는 크게 구조 모델과 행위 모델의 2 부분으로 구성된다. 구조 모델은 컴포넌트의 입/출력과 각 컴포넌트간의 관계를 기술하고 행위 모델은 컴포넌트의 동작을 표현한다.

2.1 구조 모델

구조모델은 타입과 컴포넌트로 정의할 수 있다. 타입은 일반적인 프로그래밍 언어에서 사용되는 정수, 실수, 문자열, 구조체 등으로 정의된다. 컴포넌트는 구조체의 한 종류로 입/출력을 가지며 행위 모델을 가지는 특징이 있다.

2.1.1 타입

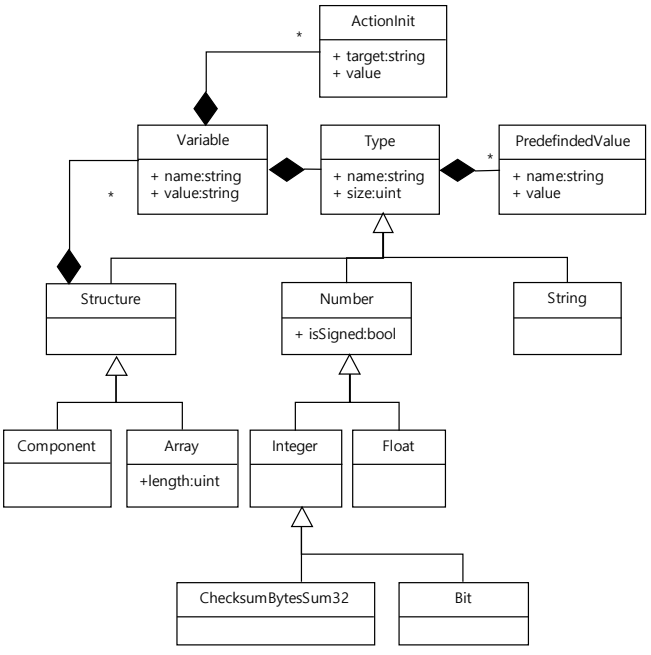


그림 1 타입 정의 (Metamodel)

타입은 대표적으로 ICD(Interface Control Document)에서 장비간의 메시지를 기술하는데 사용될 수 있다. 타입의 정의는 그림 1과 같다.

타입(Type)은 이름과 크기를 가지며, 문자열(String), 숫자(Number), 구조체(Structure) 중 하나이다. 약속값(PredefinedValue)으로 타입이 가지는 특정값에 대한 이름을 미리 정의하여 사용할 수 있으며 변수(Variable)의 타입으로 사용된다. 문자열은 문자들의 집합으로 크기가 1이면 하나의 문자를 나타낸다. 숫자는 정수(Integer) 또는 실수(Float)로 부호의 여부에 따라서 양의 값만 표현하는지를 나타낼 수 있고, 크기는 8, 16, 32, 64 중 하나로 정의되어야 한다. 구조체는 변수들의 집합이며, 컴포넌트(Component)와 배열(Array)도 구조체이다. 변수(Variable)는 이름, 타입, 값으로 구성되고 초기화 액션(ActionInit)을 이용하여 초기값을 정의할 수 있다.

표 1과 2는 타입을 표 형태로 기술하는 예제이다. 표1의 Header 구조체는 3개의 내부 변수를 가지고 있다. MsgID와 NumberOfSend는 16비트(2바이트) 크기의 양의 정수(unsigned integer)이고 WeaponType은 8비트 크기의 양의 정수이다. NumberOfSend는 3개의 약속 값을 가지며 초기값은

표 1 구조체 타입 예제

struct Header				
Byte	**Type**	**Name**	**Predefined Value**	**Init Value**
1-2	**uint16**	MsgID		
3-4	**uint16**	NumberOf Send		0
5	**uint8**	Weapon Type	0: Undefined 1: Ballistic Missile 2: Cruise Missile	Cruise Missile

표 2 구조체를 포함하는 구조체 타입 예제

struct LinkControl				
Byte	**Type**	**Name**	**PredefinedValue**	**InitValue**
1-5	Header	header		MsgID = 0x1000
6	**uint8**	Command	0: Idle 1: Link Open 2: Link Closed	1
7	**float8**	LocalTime		
8-32	**string24**	reserved		

Cruise Missile을 의미하는 2 이다.

표2의 LinkControl 구조체는 4개의 변수를 가지며 총 32바이트로 이루어져 있다. 1-5번 바이트는 표1에서 정의한 Header 이다. LinkControl 구조체에서 Header의 MsgID는 0x1000의 초기값을 가진다. 8-32번 바이트는 확장성 및 호환성을 위하여 남겨둔 필드로 문자열 타입의 변수로 정의해 놓았다.

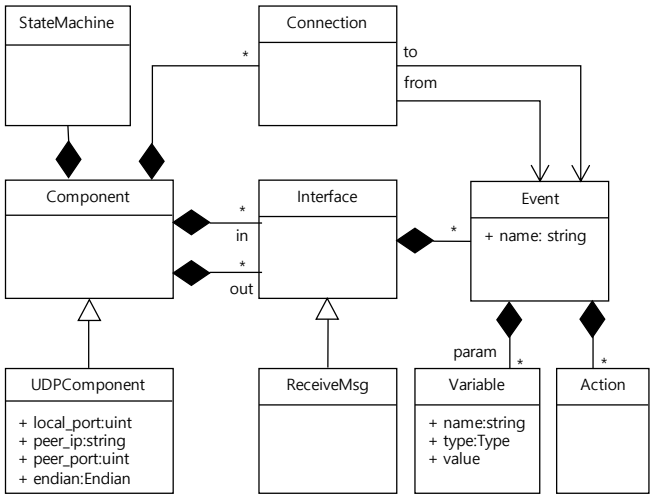


그림 2 컴포넌트 정의 (Metamodel)

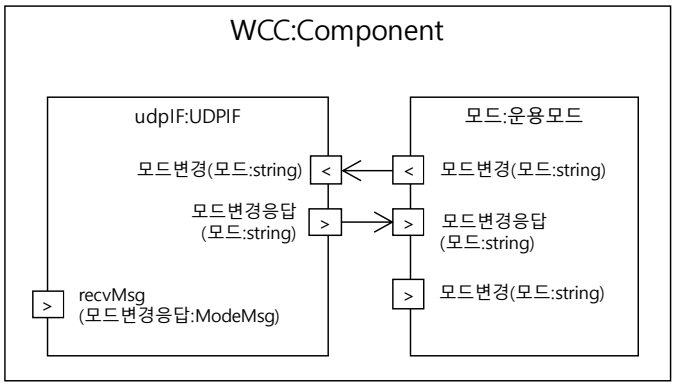


그림 3 컴포넌트 예제

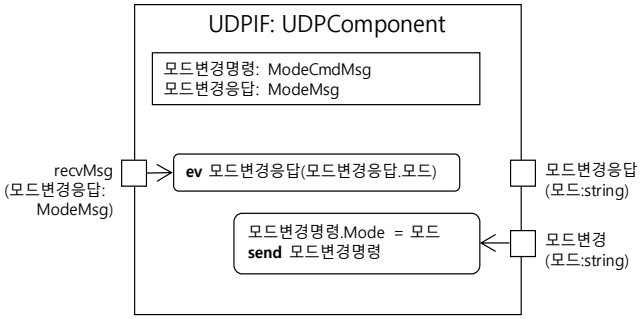


그림 4 UDPComponent 예제

2.1.2 컴포넌트

컴포넌트는 무기체계 내장형 소프트웨어 분류체계에서 CSU (Computer Software Unit) 또는 CSC (Computer Software Component)에 해당한다. 그림 2는 본 논문에서 제안하는 컴포넌트를 보여준다.

컴포넌트(Component)는 다수의 이벤트(event)로 구성된 in, out 인터페이스(Interface)와 행위를 정의하는 상태머신(StateMachine)을 포함한다. in 인터페이스는 컴포넌트의 입력 이벤트의 집합이고, out 인터페이스는 출력 이벤트의 집합이다. 이벤트는 이름, 매개변수(param), 액션으로 구성된다. 이벤트가 호출되면 액션을 수행하고, 연결(Connection)된 다음 이벤트를 호출한다. 컴포넌트는 구조체 타입의 하위 모델이므로 컴포넌트 타입의 변수를 포함한 다수의 변수들을 가질 수 있다. 또한 컴포넌트는 확장해서 사용될 수 있는데 UDPComponent가 그 예이다. UDPComponent는 UDP 통신을 수행하고, 이벤트를 상속받은 ReceiveMsg로 메시지를 입력받아서 처리할 수 있다.

그림 3과 4는 컴포넌트를 도식화한 예를 보여준다. WCC(Weapon Control Console) 컴포넌트는 UDPIF 컴포넌트 타입의 udpIF와 운용모드 컴포넌트 타입의 모드 컴포넌트를 가진다. 그림 3에서 포함 컴포넌트는 간략하게 이름, 입/출력 이벤트, 이벤트간의 연결만 표현하며, 그림 4와 같이 각 컴포넌트의 내부 구현은 따로 표현한다. 그림 4에서, UDPIF 컴포넌트는 UDPComponent로 ModeMsg 구조체 타입의 모드변경응답을 UDP 통신으로 입력받으면 모드변경응답 메시지의 모드 값을 매개변수로하여 모드변경응답 출력 이벤트를 호출한다. udpIF 컴포넌트의 모드변경응답 이벤트가 호출되면 연결된 모드 컴포넌트의 모드변경응답 이벤트가 연계적으로 호출된다. 그림 4의 ev와 send는 이벤트를 호출하는 액션과 UDP 메시지를 송신하는 액션이다.

2.1.3 행위 모델

행위 모델은 UML의 상태머신으로 기술되면 컴포넌트의 행위를 표현한다. 그림 5는 행위 모델의 메타 모델을 보여준다. 행위 모델인 상태머신(StateMachine)은 컴포넌트에 포함되며 상태(State) 하위 모델이다. 상태는 다수의 상태를 포함할 수 있고 고유의 변수와 진입(entry) 액션을 가진다. 특정 상태에는 전이(Transition)에 의해서 특정 이벤트가 호출되면 조건(Guard)이 맞을 때 액션을 수행하며 다른 상태로 바뀐다. 시간 전이(TimerTransition)는 전이의 하위 모델로 특정 시간에 실행되는 전이이다.

그림 6은 행위 모델 예제로 그림 3의 운용모드 컴포넌트 상세 설계이다. 컴포넌트의 입력 이벤트가 호출되면 전이의 이벤트가 호출되고, 전이 또는 상태의

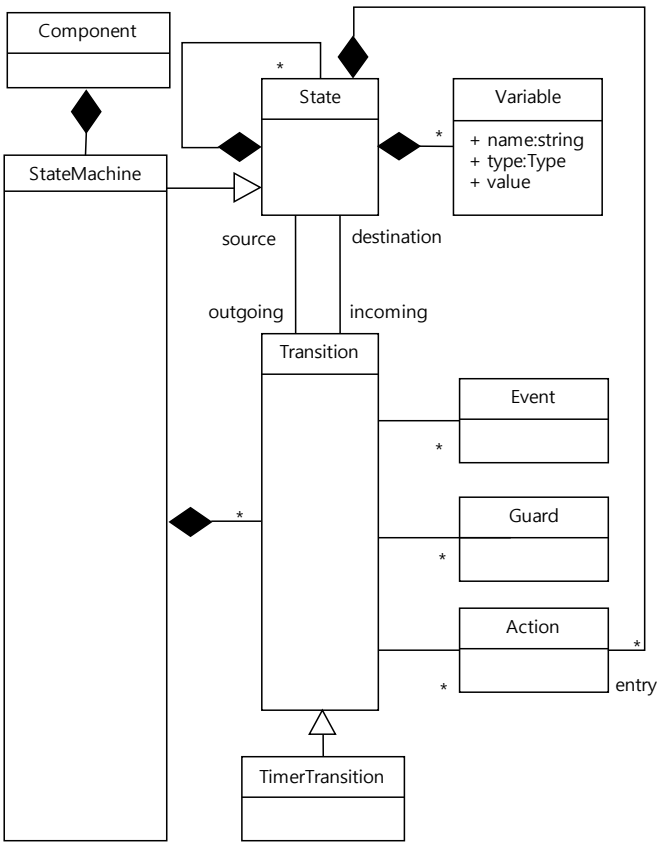


그림 5 행위모델 정의 (Metamodel)

이벤트 호출 액션을 수행하면 동일한 이름의 컴포넌트 출력 이벤트가 호출된다. 운용모드 컴포넌트의 초기 상태는 까만 작은 점이 가르키는 변경완료 상태이다. 운용모드 컴포넌트는 어느 상태에서든지 모드변경 이벤트를 받아서 처리 할 수 있으며, 모드변경 이벤트를 수신하면 변경요청모드에 수신한 모드 값을 저장하고 요청횟수를 0으로 초기화한 뒤 변경중 상태가 된다. 변경중 상태에 진입하면 진입액션(entry_ev)으로 모드변경 이벤트를 호출한다. 변경중 상태에서는 모드변경응답(모드:string), 모드변경(모드:string), 0.1초 경과의 3가지 이벤트를 처리할 수 있다. 모드변경응답 이벤트를 수신하면 현재모드의 값을 변경하고 변경완료 상태가 되고, 모드변경 메시지를 다시 수신하면 변경중 상태가 된다. 0.1초가 지나도 앞의 두 이벤트가 발생하지 않으면 요청횟수를 증가하면서 모드변경 이벤트를 재 호출하고 6회 이상 응답이 없으면 변경실패 상태가 된다.

3. 실행 엔진

2장에서 기술한 모델의 효용성을 확인하기 위하여 모델을 직접 실행하는 실행 엔진(execution engine)을 구현하였다. 실행 엔진의 역할은 모델 해석, 이더넷 또는 GUI(Graphic User Interface) 입출력 생성, 컴포넌트의 상태 머신 실행이다. GUI는 사용자와의 입출력을 담당하며 행위모델을 사용자 친화적으로 도식화하는 기능을 제공한다.

3.1 구현

실행 엔진은 어떠한 프로그래밍 언어로든 구현 가능하다. 제안한 모델의 각 요소의 기능을 올바르게 한다면 모델을 읽어서 사용자 또는 이더넷을 통해 입력받고 상태 머신을 동작시켜서 동일한 결과를 출력할 수 있다. 다만, 프로그래밍 언어의 제약이나 상태머신 처리, 이벤트 처리 등의 세부 구현 방법에 따라 실행 엔진의 성능이 달라질 수 있을 것이다.

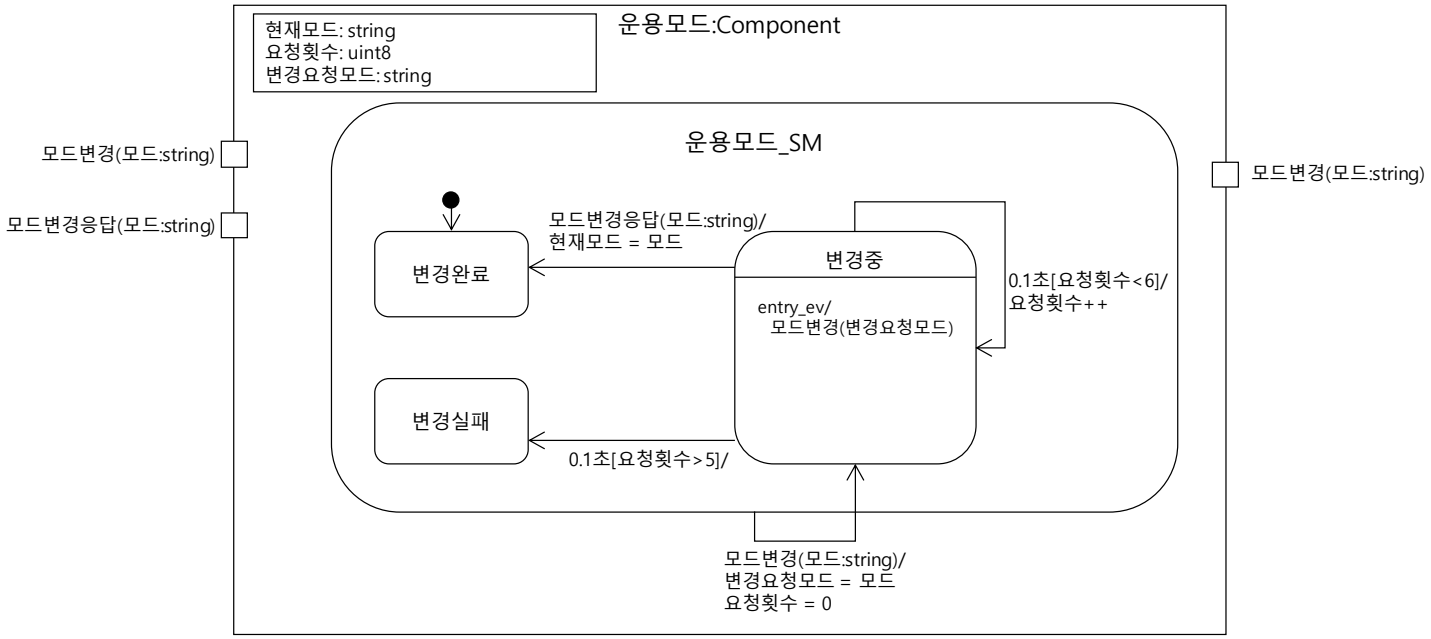


그림 6 행위 모델 예제

본 연구에서는 C++기반의 개발 플랫폼인 Qt [10]를 사용하여 실행 엔진을 구현하였다. Qt는 크로스 플랫폼 프레임워크로, 동적 라이브러리를 적용하기 때문에 한번 구현하면 윈도우, 리눅스를 비롯하여 리얼타임 OS인 VxWorks, 모바일 플랫폼인 안드로이드 등 다양한 플랫폼에서 실행할 수 있다.

이벤트 처리는 Qt의 시그날 슬롯(Signals and Slots) 프레임워크를 사용하였다. 시그날 슬롯은 타 프레임워크와 차별적인 Qt의 주요한 기능으로 객체간의 통신을 위하여 사용되는 방식이다. 기존 함수 호출 방식은 콜백(callback) 방식으로 특정 입력을 받아서 다른 함수를 수행할 때 메모리 스택을 중첩해서 쌓기 때문에 입력의 처리가 끝날 때까지 계속 메모리를 잡고 있게 된다. 또한 함수의 이름 및 매개변수가 정확하게 일치해야지 실행 가능하다는 점 등 확장성의 관점에서 다양한 문제점을 가지고 있다.

반면 시그날 슬롯 방식은 이벤트 전달 기반의 방식으로 호출, 매개변수 전달의 주요 기능에 초점을 맞추고 있다. 기존의 콜백 방식은 입력이 처리가 되기 전까지 다음 입력을 받지 못하지만 이 방식을 사용하여 이러한 제약에서 벗어났다. 또한 컴포넌트간의 호출 이벤트의 이름이 다르더라도 호출할 수 있도록 구현 가능하였다.

상태 머신의 Qt 플랫폼의 상태머신 프레임워크(State Machine Framework)를 기반으로 구현하였다. Qt의 상태머신 프레임워크는 시그날 슬롯 방식으로 구현되어 있으면 기본적인 상태와 전이를 제공한다. 본 연구에서는 이 프레임워크를 확장하여 전이조건, 액션 실행, 시간전이, 컴포넌트 입출력과의 연동 등을 구현하였다.

액션 및 조건(guard)에 사용된 연산들을 계산하기 위하여 Qt 프레임워크의 Q스크립트 엔진(QScriptEngine)을 사용하였다. 실행 엔진은 기타 모델기반 구현방식과 다르게 코드 생성 및 컴파일을 수행하지 않기 때문에 모델에서 기술하는 사칙/논리/비트 연산 등을 엔진이 수행하여야 한다. Q스크립트 엔진은 C++ 언어와 동일한 Qt 스크립트 언어를 바로 해석하여 결과를 출력하므로 기본적인 수학 연산을 실시간으로 수행하기에 적합하다.

3.2 실행 엔진 신뢰성 확보

무기체계 소프트웨어는 방사청의 무기체계 소프트웨어 개발 및 관리 매뉴얼에 따라서 정적/동적 시험을 수행하여야 한다. 정적시험은 1) 코딩규칙, 2) 실시간 오류, 3) 소스코드 메트릭 점검을 수행하여야 한다. 동적시험은 해당 CSC/CSU 수준으로 결함 영향도/발생빈도/제어가능성을 고려하여 등급을 나누고, 등급에 따라서 문장/분기/MCDC 커버리지를 적용하여 실행하여야 한다.

실행 엔진의 신뢰성 확보를 위하여 정적/동적 테스트를 수행하였다. 기존의 모델기반 개발 방식과 본 연구에서 제안하는 방식의 차이점은 코드의 생성 여부이다. 기존의 방식은 코드를 새롭게 생성하기 때문에 새로 생성된 코드에 대한 신뢰성 시험을 실시하여 신뢰성을 확보하여야 한다. 개발기간 중 또는 유지보수 기간 중 모델이 변경된다면 신뢰성 시험을 다시 실시하여야 한다. 반면, 본 연구에서 제안하는 방식은 코드를 생성하지 않고, 모델이 바뀌어도 실행 엔진의 내부 구조에는 영향을 주지 않는다. 따라서 한번 신뢰성을 확보한 실행 엔진에 대하여 추가적인 정적/동적 시험을 하지 않아도 되므로 시간과 노력을 모델의 개발에만 집중할 수 있도록 한다.

3.3 예제

제안한 모델링 언어의 효용성과 엔진의 기능을 확인하기 위하여 실제 개발 중인 무기체계의 일부 기능을 개발하였다. 예제로 개발한 기능은 해군지휘통제체계(KNCCS, Korea Naval Command Control System)와 연동하여 무장을 제어하는 기능이다. 이 기능은 이더넷으로 실제 운용중인 KNCCS 시스템과 통신하여 통신상태와 운용모드를 제어하고 임무에 따라 무장을 통제한다. 현재까지 모델링 언어와 실행엔진은 개발단계이므로, 실제 무기체계에 사용되지 않고 프로토타입으로 개발하여 가능성만을 확인하였다.

그림 7은 실행 엔진을 통하여 모델을 실행한 예제를 보여준다. 이 예제의 모델은 임무계획정보수신, 연동관리, 임무정보관리 등 11개의 컴포넌트(CSU)로 구성되어 있다. 자연스럽고 직관적인 개발을 위하여

그림 7 실행 엔진 예제

그림 8 State Combobox 예제

상태 머신을 GUI로 직접 대응하는 기능을 추가하였다. 예를 들어 State Label은 상태머신의 상태를 표시해준다. State Combobox는 그림 8과 같이 State Label의 기능에 추가로 상태머신의 입력 이벤트를 발생시킬 수 있다.

4. 결론

본 논문에서는 소프트웨어 개발 단계 중 실수 및 오류가 많이 발생하는 구현 및 구현 검증 단계를 생략하고, 설계에 집중하여 신뢰성을 높이기 위한 방법으로 실행 가능한 모델링 언어를 소개하였다. 소개한 모델링 언어의 효용성을 확인하기 위한 실행 엔진과 이를 이용하여 상부체계와 연동하는 무장통제콘솔을 개발하였다. 또한 설계를 검증하기 위한 테스트 방법을 개략적으로 논의하였다. 소개한 개발 방법의 안정적인 적용을 위해서 모델링 도구에 대한 지원과 실행 엔진의 실시간성 등의 성능 평가 및 개선을 진행 중이다. 또한 정형적이고 실행 가능할 정도로 세부적인 모델로부터 테스트 케이스를 자동으로 생성하는 연구를 수행할 예정이다.

참고문헌

[1] 방위사업청, 무기체계소프트웨어 개발 및 관리 매뉴얼, 2017. 8. 31.
[2] J. Rumbaugh, I. Jacobson and G. Booch, Unified Modeling Language Reference Manual, Pearson Higher Education, 2004.
[3] IBM, IBM Rational Rhapsody, http://ibm.com/developerworks/
[4] Programming Research Inc, QA C/C++Inc, http://prqa.com/
[5] GrammaTech, CodeSonar, http://grammatech.com/products/codesonar/
[6] Vector Software Inc, VectorCast, http://www.vectorcast.com/
[7] Liverpool Data Research Associates, LDRA, http://ldra.com/
[8] Fasoo, Sparrow QCE, http://www.fasoo.com/SPARROWQCE/
[9] Suresoft, CodeScroll, http://www.suresofttech.com/
[10] The Qt Company, Qt, http://www.qt.io/
[11] T.S. Chow, “Testing SoftwareDesign Modeled by Finite-State Machines,” IEEE Trans. Soft. Eng, vol. 3, pp. 178-187, 1978.

차량용 ECU 소프트웨어의 모델 기반 개발 시 백투백 비교 테스트 방법

김자명, 채현아 ○, 장훈*, 최병주†
이화여자대학교 컴퓨터공학과, 현대자동차*
kimjm0506@ewhain.net, coha0112@ewhain.net, hoonjang@hyundai.com*,
bjchoi@ewha.ac.kr†

Model-based testing of automotive ECU software using back-to-back comparison test

Jamyeong Kim, Hyeona Chae○, Hoon Jang*, Byoungju Choi†
Ewha Womans University, Hyundai Motor Co*

요 약

자동차 산업 분야에서 제어기의 숫자가 증가함에 따라 소프트웨어의 비중이 증가하고 있으며 기능의 다양화와 고성능화에 따라 새로운 소프트웨어 개발 프로세스를 필요로 하고 있다. 모델 기반 개발(Model Based Development)은 이러한 요구에 맞춰 자동차 제어기 소프트웨어의 새로운 개발 방식으로 떠오르고 있다. 본 논문에서는 차량용 ECU 소프트웨어의 모델 기반 개발 시 적용할 수 있는 테스트 방법 중 하나인 백투백 테스트 수행 방법을 제시하고자 한다. 이를 위해 차량용 ECU 소프트웨어의 모델 기반 개발 과정을 분석하였으며 ECU 소프트웨어의 테스트 환경을 분석하여 모델 기반 개발 과정에서의 백투백 테스트 수행 단계를 정의하였다.

1. 서 론

최근 자동차는 단순한 이동 수단을 넘어서 더욱 안전하고 편리하며, 서비스 지향, 친환경 자동차로 변화하고 있다. 이를 위해 자동차에는 수많은 ECU(전자 제어 장치, Electronic Control Unit)가 내장되었고 이 ECU들을 제어하기 위한 소프트웨어도 추가되었다. 이에 적절하게 대응하기 위해 모델 기반 개발 방식이 차량용 ECU 소프트웨어의 새로운 개발 방식으로 떠오르고 있다.

모델 기반 개발에서 주요 테스트 방법 중 하나인 백투백 비교 테스트(Back-to-back comparison test, 이하 백투백 테스트)는 자동차 기능 안전성 표준인 ISO 26262에서도 권고하고 있는 방법 중 하나이다. 하지만 백투백 테스트에 대한 구체적인 방법과 단계는 제시하고 있지 않다.

본 논문에서는 차량용 ECU 소프트웨어의 모델 기반 개발 시 백투백 테스트를 수행하는 방법을 제안한다. 이를 위해 차량용 ECU 소프트웨어의 모델 기반 개발 과정과 모델 변환 시 고려해야 할 사항을 분석하였다. 또한 모델 기반 개발 과정에서 백투백 테스트를 수행하는 단계를 정의하고 각 단계별 백투백 테스트의 특성을 분석하였다.

※ 본 연구는 2017-2018년 현대자동차 남양 연구소 산학과제 지원 하에 수행되었음.
† 교신 저자: Byoungju Choi (bjchoi@ewha.ac.kr)
○ 발표자: Hyeona Chae (coha0112@ewhain.net)

2. 모델 기반 개발

2.1. 모델 기반 개발의 개념

모델 기반 개발(Model Based Development)은 모델 기반 설계(Model Based Design)와 모델 기반 검증(Model Based Verification)을 포함하는 개발 방법으로 제품의 설계부터 검증까지 모든 단계에서 모델을 활용하며 시뮬레이션 기술을 소프트웨어 개발에 도입한 프로세스이다.

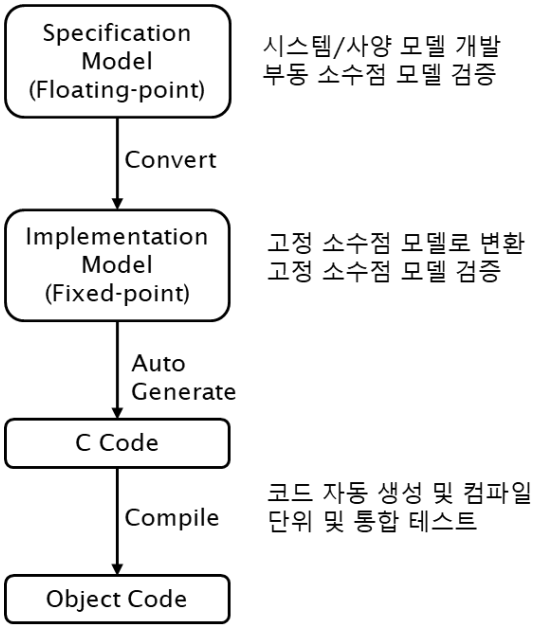
모델 기반 개발에서는 개발될 대상의 기능적 행위를 기술할 때 모델을 사용한다. 모델의 추상화 수준에 따라 시뮬레이션 또는 코드 생성에 모델을 사용할 수 있으며, 코드 중심으로 개발되었던 방식이 아닌 모델을 중심으로 개발을 한다.

기존의 개발 프로세스는 V-모델의 오른쪽 테스트 단계에 이르러서야 테스트가 가능했다. 하지만 모델 기반 개발에서는 V-모델의 왼쪽 디자인 단계의 각 절을 수행한 후 테스트를 할 수 있다. 이는 프로세스의 후반에서 크게 수정해야 하는 상황을 피하게 해준다. 모델 기반 개발 과정에서는 각 단계별 적절한 도구를 사용하여 이전 단계의 결과물을 기반으로 개발을 진행할 수 있으므로 개발 비용이 절감된다는 장점이 있다.

2.2. 고정 소수점 ECU에서의 모델 기반 개발

대부분의 차량용 ECU는 단순 연산을 지원하는 고정 소수점 방식을 지원한다. 고정 소수점 ECU를 위한 소프트웨어 개발을 할 때 부동 소수점으로부터 고정 소수점으로의 전환을 고려하는 것은 중요하다. 시스템 엔지니어는 일반적으로 알고리즘을 부동 소수점 연산으로 설계한다. 이 설계는 소프트웨어와 하드웨어의 최종 구현에 대해 크게 신경 쓰지 않고 이상적인 알고리즘 동작을 나타낸다. 그러나 대량 생산 환경에서의 소프트웨어 엔지니어 및 공급 업체는 생산 현실에 따라 이 설계를 고정 소수점으로 변환해야 하는 경우가 생긴다. [1]

모델 기반 개발은 이 프로세스에 도움을 줄 수 있다. 고정 소수점 방식의 ECU 소프트웨어의 모델 기반 개발 과정에서 설계 및 구현 부분을 간단하게 나타내면 아래의 [그림 5]와 같다. [2, 3]



[그림 1] 고정 소수점 방식의 ECU 소프트웨어의 모델 기반 설계 및 구현

본 논문에서는 MATLAB/Simulink모델을 기준으로 고정 소수점 ECU의 모델 기반 개발 과정을 살펴보도록 하겠다.

개발 대상의 초기 모델 또는 사양 모델은 일반적으로 부동 소수점 방식으로 설계된다. 이 모델은 Simulink Block Library에 있는 블록들을 사용하여 컨트롤 알고리즘을 구현한 것이다. 부동 소수점 모델을 개발한 후에는 시뮬레이션 또는 프로토타이핑 등의 방법을 통해서 모델의 알고리즘이 정확한지 확인한다. 확인이 완료된 모델은 고정 소수점 모델로 변환한다. 모델 변환을 위해서는 몇 가지 단계를 거쳐야 한다. [4, 5]

(1) 지원하지 않는 블록 교체: 먼저, Simulink의 블록 중에서 고정 소수점을 지원하지 않는 블록을 찾아서 고정 소수점을 지원하는 블록으로 대체한다. 이 단계는 연속 시간 블록을 이산 시간 블록으로 바꾸는 것도 포함한다.

(2) 블록, 데이터 타입 설정: 각 블록에 대하여 고정 소수점 데이터 타입과 스케일링(Scaling)을 설정한다. 블록의 최대값과 최소값을 설정한다.

(3) 타겟 하드웨어의 특성 설정: 모델 시뮬레이션 및 코드 생성의 결과는 타켓 하드웨어(타겟 ECU)의 특성에 따라 결정된다. char, int, long등과 다른 속성들이 특정 프로세서에 맞게 명시되어야 시뮬레이션 또는 코드 생성 시 부정확한 결과를 피할 수 있다.

(5) Rounding, Overflow 관리 방법 결정: 부동 소수점에서 고정 소수점으로 변환을 하면 허용 숫자 범위가 줄어들기 때문에 오버플로우가 발생했을 때 어떻게 처리할 것인지에 대한 방법을 결정해야 한다. 또한, 고정 소수점 숫자로 표현할 수 없는 범위의 소수점을 처리하는 방식도 결정해야 한다.

고정 소수점 모델로의 변환 후에는 변환한 모델을 검증한다. 고정 소수점 모델의 시뮬레이션 결과가 부동 소수점 모델의 결과와 비교했을 때 수용 가능한 범위 안에서 차이가 났는지 확인한다. 만약 두 모델의 차이가 허용 가능 범위를 벗어났다면 고정 소수점 모델로의 변환 과정으로 돌아가서 설정을 변경하여 재 수행하거나 필요에 따라 부동 소수점 모델로 돌아간다.

고정 소수점 모델이 검증된 후에는 그 모델을 기반으로 코드를 생성한다. 코드 생성 후에는 단위 및 통합 테스트를 수행한다. 코드는 데스크탑과 같은 호스트 머신 상의 환경에서 테스트된다. 또한 고정 소수점 모델과 자동 생성된 코드 간의 백투백 테스트도 수행한다.

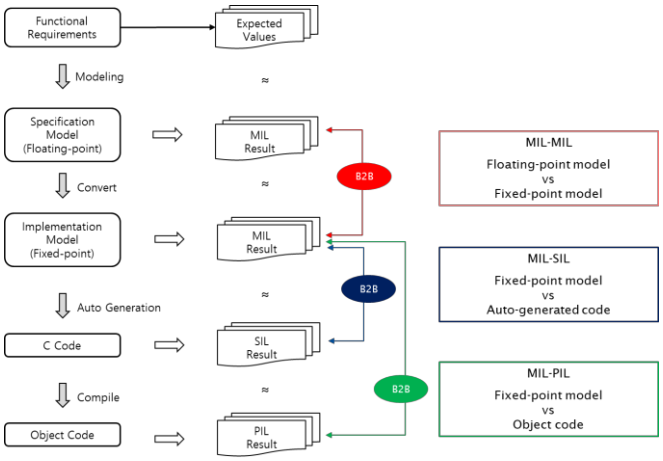
오브젝트 코드(Object code)는 타겟 하드웨어 또는 에뮬레이터에서 테스트 된다. 이 코드도 마찬가지로 고정 소수점 모델과의 백투백 테스트를 수행해야 한다.

3. 차량용 ECU 소프트웨어의 백투백 테스트

백투백 테스트는 두 개의 또는 그 이상의 테스트 대상 시스템에 대하여 각각 동일한 입력 값을 주고 실행하여 그 결과값을 비교 한 후, 불일치 할 경우에 그 불일치를 서로 비교 분석하는 비교 시험의 일종이다. [6] 백투백 테스트 수행 시에는 테스트 대상과 테스트 환경에 맞는 테스트 케이스 도출이 필요하다. 테스트 케이스는 요구사항으로부터 도출할 수도 있고, 기존 시스템이나 시스템 모델의 측정을 통해서도 얻을 수 있다. 유의미한 테스트 케이스 도출을 위해 테스트 대상에 따라 차이가 생길 수 있는 부분을 고려하여야

한다.

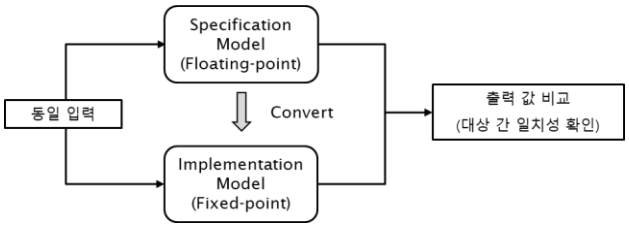
[그림 2]은 고정 소수점 방식의 ECU 소프트웨어를 개발할 때, 백투백 테스트를 수행하는 단계를 보여준다. 고정 소수점 방식을 사용하는 차량용 ECU의 소프트웨어를 모델 기반 개발 방식으로 개발한다면 부동 소수점 모델과 고정 소수점 모델 간의 MIL-MIL 백투백 테스트, 고정 소수점 모델과 자동 생성된 코드 간의 MIL-SIL 백투백 테스트, 고정 소수점 모델과 오브젝트 코드 간의 MIL-PIL 백투백 테스트를 수행할 수 있다.



[그림 2] 고정 소수점 방식의 ECU 소프트웨어의 백투백 테스트

3.1. MIL-MIL 백투백 테스트

MIL-MIL 백투백 테스트는 모델과 모델 사이의 백투백 테스트이다. 차량용 ECU 소프트웨어의 모델 기반 개발 과정에서 보면 부동 소수점 모델과 고정 소수점 모델이 그 대상이 될 수 있다. [그림 3]은 MIL-MIL 백투백 테스트의 개념을 보여준다.



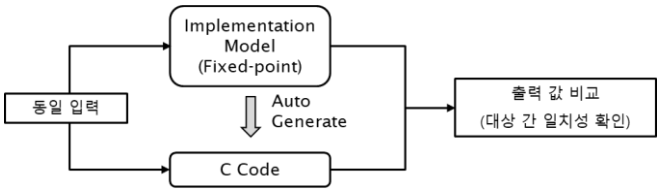
[그림 3] MIL-MIL 백투백 테스트

백투백 테스트를 위해서는 두 대상간 차이가 생길 수 있는 테스트 케이스를 도출해 내는 것이 중요하다. MIL-MIL 백투백 테스트에서는 부동 소수점 모델을 검증하기 위해 사용했던 테스트 데이터를 활용할 수 있다. 이 경우에 테스트 케이스는 요구사항으로부터 도출될 수 있지만, 기존의 시스템을 측정함으로써 얻을 수도 있고, 임베디드 소프트웨어와 상호 동작하는 물리적 시스템의 모델을 통해서도 얻을 수 있다. 이외에도 MIL-MIL 백투백 테스트를 위한 테스트 케이스로 부동 소수점 모델의 허용 범위 이내이며 고정 소수점 모델의 범위를 벗어나는 입력 값 또한 사용할 수 있다. 반올림이나 올림과 같은 Rounding 방법에 따라 결과가 달라지는 입력 값도 MIL-MIL 테스트 케이스에 적합하다.

이러한 테스트 케이스들로 각각 부동 소수점 모델과 고정 소수점 모델에서 테스트를 수행한 뒤, 동일한 결과가 나오는지 수행 결과를 비교한다. 수행 결과가 오차 허용 범위 이내이면 백투백 테스트는 패스되었다고 볼 수 있다. 만일 수행 결과에 차이가 있다면 결함 가능성이 있다는 것을 의미하므로 테스트 대상 모델을 재검토하고 필요에 따라 모델을 수정한다. 모델 수정 후에는 테스트를 다시 수행해야 한다.

3.2. MIL-SIL 백투백 테스트

MIL-SIL 백투백 테스트는 모델과 코드 사이의 백투백 테스트이다. 고정 소수점 모델과 그 모델로부터 자동 생성한 코드가 그 대상이 된다. 참고로 자동 생성하지 않고 모델링한 설계를 토대로 직접 작성하였을 경우도 해당된다. 본 논문에서는 자동생성코드의 경우로 기술하겠다. [그림 4]는 MIL-SIL 백투백 테스트 방법을 보여준다.



[그림 4] MIL-SIL 백투백 테스트

MIL-SIL 에서도 백투백 테스트를 수행을 다시해야하는 이유는 부동 - 고정 소수점 변환뿐만 아니라, 도구에서 모델로부터 코드를 자동생성하는 과정에서의 코드 셋팅 등 오류가 있을 수 있기때문이다.

MIL-SIL 백투백 테스트에서도 부동 소수점 모델 검증 시 사용한 테스트 케이스를 활용할 수 있다. SIL 시험에서도 MIL 테스트에서 사용된 테스트 데이터와 모델 인프라 구조가 재사용된다.

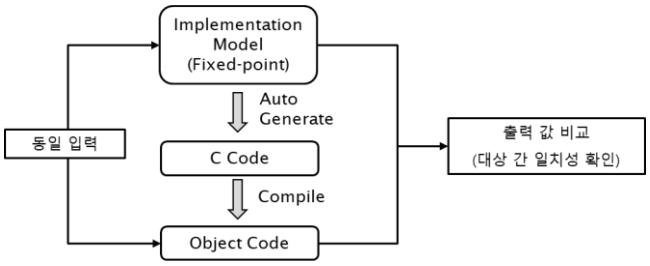
MIL-MIL 백투백 테스트와 마찬가지로 MIL-SIL 백투백 테스트에서도 각각 고정 소수점 모델과 자동 생성된 코드의 실행 결과의 차이가 허용 범위 이내이면 백투백 테스트를 패스한 것이고, 허용 범위를 벗어나는 차이가 있을 때는 테스트 대상을 재검토하여 수정해야 하고 테스트를 재 수행한다.

3.3. MIL-PIL 백투백 테스트

MIL-PIL 백투백 테스트는 모델과 오브젝트 코드 사이의 백투백 테스트이다. 차량용 ECU 소프트웨어의 모델 기반 개발 과정에서 보면 고정 소수점 모델과

자동 생성된 코드를 컴파일 한 오브젝트 코드가 테스트 대상이 될 수 있다. 이 경우 컴파일 과정과 반올림 과정에서 문제가 있을 수 있다.

[그림 5]는 MIL-PIL 백투백 테스트 방법을 보여준다. PIL 환경에서의 테스트는 개념적으로 SIL 환경에서의 테스트와 유사하다. 따라서 MIL-PIL 백투백 테스트에서도 MIL-SIL 백투백 테스트에서 사용한 테스트 케이스를 사용하여 테스트 할 수 있다. MIL-PIL 백투백 테스트의 경우, 고정 소수점 모델과 자동 생성한 코드를 컴파일 한 오브젝트 코드가 테스트 대상이다.



[그림 5] MIL-PIL 백투백 테스트

MIL-PIL 백투백 테스트에서도 각각 고정 소수점 모델과 오브젝트 코드의 실행 결과의 차이가 허용 범위 이내이면 백투백 테스트를 패스하였다고 본다. 만약 그 차이가 허용 범위를 벗어나는 경우에는 테스트 대상을 재검토하여 수정해야 하고 테스트를 다시 수행한다.

4. 결론

본 논문에서는 차량용 ECU 소프트웨어의 모델 기반 개발 과정을 분석하고, 백투백 테스트를 수행하는 단계에 대해서 정의하였다.

메모리 효율성, 낮은 가격과 크기, 적은 전력 소모량 등의 장점이 있기 때문에 차량용 ECU는 대부분 고정 소수점 방식의 프로세서를 사용한다. 하지만 시스템 레벨에서의 설계 및 알고리즘은 대개 부동 소수점 방식으로 작성된다. 따라서 이 사양 모델을 타겟에 맞는 고정 소수점 방식으로 변환하는 과정이 필요하다. 설계에 대한 검증이 완료된 부동 소수점 방식의 사양 모델은 몇 가지 단계를 거쳐 고정 소수점 방식의 구현 모델로 변환한다. 변환된 고정 소수점 모델로부터 코드를 자동 생성하고 이 코드를 컴파일하여 오브젝트 코드를 생성한다.

이 과정에서 모델 및 코드에 대한 설계 검증과 단위 및 통합 테스트도 수행된다. 모델 기반 개발에서는 모델에 대하여 검증 및 테스트가 가능한데, 이때 반드시 모델과 코드 간의 백투백 테스트가 수행되어야 한다. 고정소수점 방식 ECU 소프트웨어의 모델 기반 개발에서 백투백 테스트는 크게 3가지로 나눌 수 있다. MIL-MIL 백투백 테스트, MIL-SIL 백투백 테스트, MIL-PIL 백투백 테스트가 그것이다. 각 단계별 테스트 대상과 유의미한 테스트 케이스를 정하여 백투백 테스트를 수행하여야 한다.

향후 연구에서는 차량용 ECU 소프트웨어의 백투백 테스트를 위한 효율적인 테스트 케이스 도출 방안이 마련되어야 한다. 또한 본 논문에서 다룬 차량용 ECU 소프트웨어의 모델 기반 개발 과정과 백투백 테스트 방법에 대한 사례 연구가 필요하다.

5. 참고문헌

[1] Anoop, C. V., and C. Betta. "Comparative Study of Fixed-Point and Floating-Point Code for a Fixed-Point Micro." Proceedings of the dSPACE User Conference. 2012.

[2] Beine, M. "A model-based reference workflow for the development of safety-critical software." Embedded Real Time Software and Systems 1-6. (2010):

[3] Conrad, T. Erkkinen, et al. "Code Generation Verification-Assessing Numerical Equivalence between Simulink Models and Generated Code." 4th Conference Simulation and Testing in Algorithm and Software Development for Automobile Electronics. 2010.

[4] Erkkinen, Tom. Fixed-Point ECU Development with Model-Based Design. SAE Technical Paper, No. 2008-01-0744. 2008

[5] Mathworks, Tips for Fixed-Point Modeling and Code Generation for Simulink 7 (R2008a). [Online] Available:https://kr.mathworks.com/matlabcentral/fileexchange/19835-tips-for-fixed-point-modeling-and-code-generation-for-simulink-7-r2008a.

[6] 한무희, 이승곤, and 이창호. "모델과 소스코드 간 백투백 테스트 접근법 및 적용 사례 연구." 한국자동차공학회 추계학술대회 및 전시회 1713-1716. (2011):

오픈소스 소프트웨어 저장소에서 결함 수정 코드 수집 방안

임성빈, 양수진○, 남재창

한동대학교 전산전자공학부

21700613@handong.edu, 21500394@handong.edu, jcnam@handong.edu

Mining Fix Patches In Open Source Software Archive

Seongbin Lim, Sujin Yang○, Jaechang Nam

School of Computer Science and Electronic Engineering, Handong Global University

요 약

소프트웨어 품질 관리를 위해 개발중인 프로그램의 소스코드(Source Code)에서 결함을 발생시키는 원인을 찾고 분석하는 것은 중요하다. 이를 위해 과거의 수많은 프로젝트에서 결함을 발생시켰던 소스코드와 결함을 수정한 코드를 수집해야 한다. 충분한 양의 결함 데이터가 수집되면, 결함을 예측하는 기계학습 모델의 학습 데이터로 사용하거나 소프트웨어 품질 보증을 위한 다양한 응용도구 개발에 사용될 수 있다. 본 연구에서는 소프트웨어 저장소에 존재하는 결함을 수정하는 소스코드(fix patch)와 관련 정보를 자동으로 수집하는 도구를 개발하였다. 오픈소스 프로젝트 개발 이력이 저장되는 주요 소프트웨어 저장소에서 결함의 정보를 자동으로 수집하는 본 도구가, 추후 소프트웨어 품질 관리에 필요한 다양한 응용 도구 연구에 도움이 되길 기대한다.

1. 서 론

현재 소프트웨어 산업에서 개발자들은 항상 수많은 결함에 맞닥뜨리고 있다. 이에 대한 명확한 해결 방안이 없는 와중에 정부의 코딩 교육 강화 정책으로 인해 앞으로 소프트웨어 개발자가 더욱 늘어날 전망이다[1, 2]. 따라서 개발자들이 결함을 찾고 프로그램에 안정성을 높일 수 있도록 도와주는 소스코드 품질 분석 프로그램의 필요성이 나날이 높아지고 있다. 그러나 실제 산업 현장에서 사용할 수 있는 프로그램은 여전히 많은 제한점들이 있다[3]. 이는 곧 대부분의 개발자들이 직접 결함을 고치기 위해 혹은 발견하지 못한 결함을 찾기 위해 프로젝트에 많은 시간과 비용을 소비한다는 것을 의미한다. 실제로 소프트웨어를 개발할 때에 50~90% 정도의 비용이 소프트웨어의 유지 및 보수를 위해 사용한다[3]. 이러한 개발자들의 고통을 덜어주고, 지속적인 소프트웨어 품질 관리를 위하여 신뢰성 있는 결함 예측과 검출 기술의 발전이 필요하다.

결함 예측과 검출 기술은 현존하는 결함을 분석하면서부터 시작된다. 이를 위해서 상당한 양의 결함 정보가 필요 하다. 기존의 결함 예측 논문은 직접 새로운 결함 수집 도구를 만들어 결함을 연구했다[4]. 혹은 이미 공개된 결함 정보를 사용한다[5, 6]. 공개된 결함 정보에 만족하지 못해 기존의 정보를 재가공하여 사용하기도 한다[7]. 이러한 기존의 결함 예측 논문은 결함 정보를 활용한 연구에 초점이 맞춰져 있고 이미 수집한 결함 정보만을 공유하는 경우가 대부분이다. 일부 결함 정보 수집 방안을 제시한 연구가 있기는 하나 수집 도구의 설계와 구현의 논의만 되어 있고 실제 사용할 수 있는 도구는 공유되어 있지 않거나 최신 소프트웨어 저장소에서 적용하기 어렵다[8, 9]. 본 연구에서는 결함 분석을 위해 다양한 오픈 소스 프로젝트 저장소에 접근하여 결함 정보를 자동으로 수집하는 도구를 설계하고 공개하였다.

2. 결함 정보 수집 도구 설계

2.1 결함 정보 수집

깃(Git)은 프로그램을 개발할 때 프로젝트의 관리를 도와주는 소프트웨어로 분산 버전 관리 시스템이다 [10]. 프로젝트가 수정될 때마다 변경되는 모든 정보가 버전 관리 시스템을 통해 기록으로 남는다. 따라서 프로젝트 내부에 문제가 일어났을 때 이전 시점으로 되돌아가 수정 내용을 비교하거나 문제의 원인을 추적할 수 있다. 마찬가지로 깃이 적용된 프로젝트에서 소스 코드가 변경될 때마다 기록이 모두 남고 이러한 기록의 가장 작은 단위가 커밋(Commit)이다. 규모가 큰 프로젝트에는 과거로부터 쌓인 수많은 커밋들이 존재하기 때문에 결함과 관련 있는 커밋과 결함과 관련 없는 커밋들이 함께 섞여 있다. 결함과 관련있는 정보를 찾기 위해서는 다수의 커밋들 중에서 결함과 관련있는 커밋을 추출해야 한다. 본 연구에서 개발한 도구는 오픈 소스 프로젝트에서 보편적으로 사용되는 소프트웨어 저장소인 깃허브(GitHub)와 지라(Jira)에서 결함과 관련된 정보를 수집할 수 있도록 고안되었다.

소프트웨어 프로젝트를 진행하면서 개발자들은 협업을 위해 서로 프로젝트의 개선점, 해결해야 하는 결함, 또는 해야 할 일들과 같은 것들을 의논하고 해결하여야 한다. 이렇게 개발자들 사이에서 의논되고 해결되어야 하는 주제를 이슈(Issue)라고 한다. 이러한 수많은 이슈들은 지라와 깃허브 같은 소프트웨어 저장소를 통하여 관리된다.

이러한 소프트웨어 저장소에서 결함 정보를 수집하는 방법은 다음과 같다.

2.1.1 깃허브 이슈에서 결함 정보 수집

깃허브는 가장 인기 있는 오픈소스 코드 저장소이다. 분산 버전 관리 도구인 깃을 사용하기 때문에 여러 사람이 함께하는 협업 프로젝트를 관리하기 매우 용이하다. 이와 같은 이유로 깃허브 안에는 수많은 프로젝트의 커밋 데이터들이 저장되어 있다.

깃허브는 효율적인 프로젝트 관리를 위해 여러 기능을 지원해 주고 있다. 그중 하나가 이슈다. 이슈는 프로젝트에서 작업을 하거나 해결을 해야하는 작업 내용들 담고 있다. 대표적으로 버전 업그레이드, 결함의 발견과 해결, 개선해야 할 기능, 논의가 필요한 부분과 같은 것들이 있고 각각의 주제에 맞는 레이블(Label)을 부여해 줌으로써 관련된 주제에 접근을 용이하게 해준다. 또한 해당 주제가 논의가 필요한 상태이면 열림(Open), 문제가 해결된 상태이면 닫힘(Closed)으로 구분된다. 본 연구에서 개발된 도구는 깃허브에서 관리하는 오픈소스 프로젝트 중 상태가 닫힘으로 되어있고 레이블이 버그인 깃 커밋을 수집하는 도구이다. 결함을 정확하게 분석하기 위해서는 완전히 해결된 결함만을 필요로 하기 때문이다.

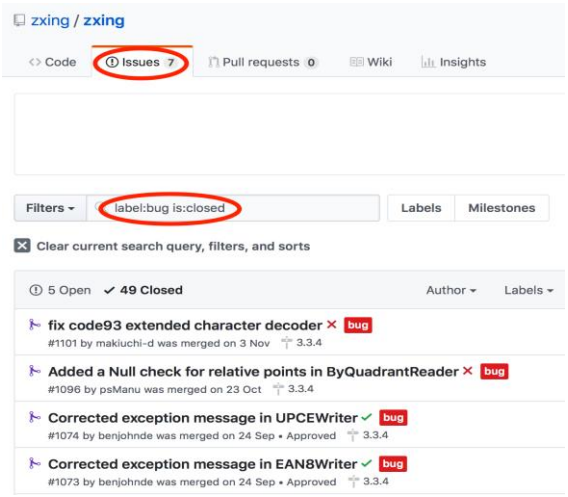


그림 1. 깃허브의 이슈 화면 예제

그림 1은 깃허브에서 찾을 수 있는 오픈소스 프로젝트이다. 이 프로젝트의 이슈 페이지에 들어가 레이블을 해결된 버그로 설정하니 해당되는 데이터가 49개 라고 나온다. 즉 해결된 버그 이슈가 49개 있다는 것이다. 이처럼 깃허브에서는 이슈 페이지를 이용하여 프로젝트의 버그 수정 내역에 쉽게 접근할 수 있다.

2.1.2 지라 이슈에서 결함 정보 수집

지라는 이슈를 관리하기 위해 개발된 전문적인 소프트웨어로써 협업 프로젝트에서 필요한 계획 세우기, 분업 정리, 보고서 관리와 버그 추적과 같은 다양한 기능을 개발자에게 제공해준다[11]. 지라는 각각의 이슈 성격을 분명하게 나타내는 레이블과 이슈의 진행 상황을 나타내는 상태(Status)를 붙여서 관리한다. 레이블의 종류로는 업무(Task), 테스트(Test), 버그(Bug), 스토리(Story) 등등이 있고, 상태의 종류로는 열림(Open), 닫힘(Closed), 해결(Resolved) 등등이 존재한다.

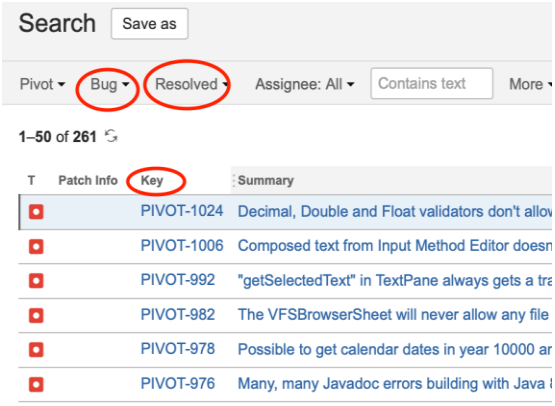


그림2. 지라의 이슈 화면 예제

그림 2와 같이 지라에서는 버그 레이블이 붙은 이슈들만 따로 검색할 수 있으며, 그 정보를 csv 형식의 파일로 다운로드할 수 있다. 또한 지라는 모든 이슈를 고유한 식별 인자인 키(Key)를 붙여서 관리하고 대부분의 프로젝트는 모든 커밋 제목에 관련된 이슈 키를 포함한다. 따라서 주어진 조건을 통해 얻은 파일에서 있는 이슈 키가 임의의 커밋의 제목에 존재한다면 그 커밋은 결함과 관련되었다고 식별할 수 있다.

2.1.3 깃 커밋 메시지에서 결함 정보 수집

개발자들이 소프트웨어 저장소를 통해 공유하는 정보를 이용하는 방법 이외에도 커밋을 할 때마다 생기는 커밋 메시지(Commit Message)를 통해 결함이 있는 커밋인지 추론할 수 있다. 개발자들은 소스코드를 수정 후 커밋을 할 때 커밋 메시지를 작성한다. 보통 커밋 메시지는 코드의 유지 보수와 협업을 더 쉽게 하기 위해 중요한 키워드를 포함하여 작성하도록 암묵적 동의가 되어있다[12]. 본 연구는 관련연구들에서 보편적으로 사용하는 방법대로 이 특징을 이용하여 커밋 메시지에 'fix'나 'bug', 'resolved'의 단어를 포함하는 경우 해당 커밋은 결함과 관련되었다고 식별하였다[8].

2.2 결함 내용 출력

깃 커밋에서는 커밋이 가지고 있는 모든 정보에 접근할 수 있다. 여기서 모든 정보에는 커밋한 날짜, 저자 이름, 저자 이메일, 커밋 메시지, 수정한 소스파일 경로, 소스코드 패치(Patch)가 포함된다. 패치란 이전 시점의 커밋과 비교했을 때에 변경된 소스코드와의 텍스트 차이로 그림 3은 패치의 한 예시이다. 본 연구에서는 자료의 질을 높이기 위해 커밋의 정보 중에 개발자의 이메일과 같이 분석에 필요 없는 부분을 제외하고 결함이 존재하는 데이터에서 유용한 부분만 수집하여 사용자가 원하는

표1. 결함 정보 수집 결과

Project	ShortMessage	Commit Hash	Date	Author	Path	Patch
pivot	Resolve PIVOT-61	1ec074aa5348f	2010.08.27 00	Gregory K. Brov	wtk/src/org/ap	diff --git a/wtk/src/org/apache/pivot/ index d239f26..a53c5fa 100644 --- a/wtk/src/org/apache/pivot/wtk/F +++ b/wtk/src/org/apache/pivot/wtk/ @@ -621,4 +621,8 @@
pivot	PIVOT-946: Add t	a033f6ec685fc0	2014.04.22 09	Roger Lee Whit	wtk/src/org/ap	diff --git a/wtk/src/org/apache/pivot/ index 7d3247f..1930814 100644 --- a/wtk/src/org/apache/pivot/wtk/s +++ b/wtk/src/org/apache/pivot/wtk/ @@ -1348,7 +1348,7 @@
pivot	Fix bug in DropSh	4fe9771bc59b0	2009.04.16 03	Gregory K. Brov	demos/src/piv	diff --git a/demos/src/pivot/demos/fi index fa7b00c..3eed657 100644 --- a/demos/src/pivot/demos/filebrov +++ b/demos/src/pivot/demos/filebr

결과를 csv 형식의 파일로 출력한다. 결과를 출력하는 파일은 표1과 같이 프로젝트(Project), 커밋 메시지, 깃 커밋 해시(Hash), 날짜(Date), 저자 이름(Author), 파일 경로 (Path), 패치(Patch)의 내용으로 구성되어 있다.

```
From 3a6891d268c4a23ebe37a51720917f1cfb606dad Mon Sep 17 00:00:00 2001
From: Sean Owen <srowen@gmail.com>
Date: Fri, 7 Dec 2018 07:04:40 -0600
Subject: [PATCH] Clarify CONTRIBUTING.md

---
 .github/CONTRIBUTING.md | 19 +++++++------------
 1 file changed, 7 insertions(+), 12 deletions(-)

diff --git a/.github/CONTRIBUTING.md b/.github/CONTRIBUTING.md
index d065b9f1b..5f771722b 100644
--- a/.github/CONTRIBUTING.md
+++ b/.github/CONTRIBUTING.md
@@ -1,9 +1,9 @@
 ## Contributing

 The project is in maintenance mode, meaning, changes are driven by contr
-Only bug fixes and minor enhancements will be considered. The Barcode Sc
-no longer be published, so it's unlikely any changes will be accepted fo
-There is otherwise no active development or roadmap for this project. It
+Only bug fixes. The Barcode Scanner app can
+no longer be published, so almost no fixes will be accepted for it.
+There is otherwise no development or roadmap for this project.
```

그림 3. 패치 예시

패치를 출력할 때에는 각각의 패치에서 변경된 라인의 수, 즉 패치의 크기(Patch size)도 고려해야 한다. 일반적으로 결함을 해결한 커밋 패치에서 지워진 부분이 과거에 결함을 유발한 소스코드이고, 변경되거나 추가된 부분이 결함을 해결하는 소스코드로 간주된다. 만약 결함을 수정한 커밋의 패치에서 지워진 라인이 너무 많다면 어떤 부분이 이전에 결함을 유발했는지 분석하기 어려울 것이다. 반대로 추가된 라인이 너무 많다면 어떤 부분이 결함을 해결했는지 분석하기 어려울 것이다. 혹은 소스코드를 재정렬하거나 주석을 새로 다는 것과 같이 불필요한 정보인 경우도 많다. 이처럼 크기가 큰 패치를 분석이 까다롭다. 또 이전에 존재했던 연구들에 따르면 대부분의 결함을 수정한 패치는 짧게 수정하는 비율이 높은 경향이 있다 [13]. 이러한 점을 고려하여 본 연구는 용이한 패치 분석을 위해 다양한 패치 사이즈 기준을 설정할 수 있는 기능을 추가하였다.

3. 결과

표 1은 개발한 소프트웨어를 이용해 이름이 pivot인 프로젝트를 대상으로 얻은 결과 표이다.

표2는 결함 정보 수집 도구를 실제 오픈소스 프로젝트 소프트웨어 저장소에서 적용한 결함 수정 패치 수집 결과를 보여준다. 표2에서 추출 저장소는 결함 데이터를 얻어낸 오픈소스 저장소를 의미한다. 프로젝트 열은 해당 프로젝트의 이름을 의미한다. 전체 커밋의 개수는 프로젝트가 현재까지 진행되면서 쌓인 커밋의 개수를 의미하고 수집 시간은 소프트웨어가 수집하는데 걸린 시간이다. 데이터 수집을 위해 2.3 GHz Intel Core i5와 RAM 8GB가 탑재된 장비를 이용하였다. 수집된 패치의 개수는 결함을 해결한 커밋에서 나온 패치의 총 개수이고 크기가 5 이하인 패치만 추출하였다. 커밋에 기여한 사람의 수는 프로젝트를 개발 중에 모든 기여한 개발자의 숫자이다.

표2. 다양한 프로젝트의 수집 결과

추출 저장소	프로젝트	전체 커밋의 개수	수집 시간	수집된 결함 패치 개수 (patch size <= 5)	커밋에 기여한 개발자 수
지라	pivot	5150	246s	785	16
지라	brooklyn	13401	91s	415	111
깃허브	zxing	3480	286s	33	90
깃허브	Android-Universal-Image-Loader	1025	516s	32	35

표2에서 관찰할 수 있는 특이점은 지라에서 수집한 brooklyn의 전체 커밋의 개수가 pivot보다 더 많지만 수집하는 시간은 더 오래 걸렸다는 점이다. 커밋에서 패치를 분석하는 시간이 전체 커밋에서 결함 패치 커밋을 찾는 시간 보다 더 많이 걸리는 것이 가장 큰 이유이다. 표2에서 brooklyn은 785개 결함 패치가 pivot은 415개의 패치가 분석이 되었다. 지라에서 결함 정보를 수집하는 것과 깃허브에서 수집하는 시간의 차이도 크게 관찰되었다.

왜냐하면, 깃허브에서 결함 정보를 추출할 때는 원격으로 이슈 정보를 수집하기에 네트워크 속도에 영향을 받는다. 그래서 표2에 나온 깃허브에 있는 오픈소스 프로젝트의 수집된 패치가 33, 32개임에도 불구하고 수집 시간은 지라에서 수집하는 시간보다 더 많은 시간이 소요되었다.

4. 결론 및 향후 계획

본 연구는 소프트웨어 분산 관리 도구인 깃을 이용한 오픈 소스 프로젝트에서 결함과 관련된 정보들을 추출하여 활용성 높은 자료 제작을 목표로 하였다. 지금까지는 결함 예측과 추출을 위해서 소프트웨어 품질 담당 전문가가 직접 결함이 있는 데이터를 수집해야 하는 불편함이 있었다. 그러나 본 연구에서 자동으로 결함 정보의 수집이 가능한 도구를 구현하여 이러한 불편함을 해소하였다.

설계 방법은 오픈 소스 소프트웨어 저장소인 깃허브와 지라의 이슈를 추출하여, 해결된 결함을 csv 형식의 파일로 출력하였다. 따라서 자료를 읽는 개발자가 결함 정보를 수월하게 분석 할 수 있다. 예를 들어 개발자가 직접 결함이 있는 소스코드를 읽으며 결함을 유발하는 특정 메트릭을 찾거나 결함이 있는 소스코드의 특징을 분석하여 메트릭으로 만들 수 있다. 이렇게 개발자가 찾아낸 정보는 소프트웨어 결함 예측을 위한 기계학습 모델을 만드는데 사용되거나 결함 검출 도구 설계에 활용될 수 있다.

향후 연구에는 결함이 해결된 소스코드를 출력하는 현재 상태를 발전시켜, 결함을 유발하는 소스코드가 유입된 시점의 커밋 해시와 소스코드를 수집할 수 있는 기능을 추가할 예정이다. 본 연구를 통해 개발한 도구, BugPatchCollector는 다음에서 다운로드 받을 수 있다. https://github.com/HGUISEL/BugPatchCollector

※ 본 연구는 과학기술정보통신부와 정보통신기술진흥센터의 2019년도 소프트웨어 중심대학 지원사업(2017-0-00130)과 2018년도 정부 (과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2018R1C1B6001919).

V. 참고 문헌

[1] "「소프트웨어 교육 활성화 기본계획」 발표", 교육부, Retrieved from: http://www.moe.go.kr/boardCnts/view.do?boardID=294&boardSeq=65033&lev=0&searchType=null&statusYN=W&page=4&s=moe&m=0503&opType=N

[2] "소프트웨어 교육 운영 지침", 교육부, Retrieved from:http://webcache.googleusercontent.com/search?q=cache:f_Jb_dkH1IJ:www.unjeong.es.kr/wah/main/file/download.do%3Fid%3D9969_1+&cd=2&hl=ko&ct=clnk&gl=kr&client=safari

[3] MD.SHAHADAT HOSSAIN, "CHALLENGES OF SOFTWARE QUALITY ASSURANCE AND TESTING", IJSECS, Volume 4 Issue 1, pp. 133-144, 2018

[4] X Yang, D Lo, X Xia, Y Zhang, and J Sun, "Deep Learning for Just-In-Time Defect Prediction " QRS '15 Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security, 17-26

[5] J Nam, S Wang, Y Xi and L Tan, "Poster: Designing Bug Detection Rules for Fewer False Alarms" ICSE '18 Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, 315-316

[6] T Fukushima, Y Kamei, S McIntosh, K Yamashita, and N Ubayashi, "An Empirical Study of Just-in-Time Defect Prediction using Cross-Project Models", MSR 2014 Proceedings of the 11th Working Conference on Mining Software Repositories, 172-181

[7] F Zhang, A Mockus, I Keivanloo and Y Zou, "Towards Building a Universal Defect Prediction Model", MSR 2014 Proceedings of the 11th Working Conference on Mining Software Repositories, 182-191

[8] J Śliwerski, T Zimmermann and A Zeller, "When Do Changes Induce Fixes?", MSR '05 Proceedings of the 2005 international workshop on Mining software repositories, 1-5

[9] A Mockus, "Amassing and indexing a large sample of version control systems: towards the census of public source code history", 2009 6th IEEE International Working Conference on Mining Software Repositories, 11-20

[10] "About - Git", Git Official Site, accessed Dec 21, 2018,Retrieved from: https://git-scm.com/about/info-assurance

[11] Jan Godlewski, "The Top 6 Benefits of Using Time Tracking with Jira", Retrieved from: https://www.timecamp.com/blog/2015/09/top-6-benefits-using-time-tracking-jira-2/

[12] "Standard for Git commit messages" LimeSurvey. last modified Mar 1. 2016, accessed accessed Dec 21, 2018, Retrieved from: https://manual.limesurvey.org/Standard_for_Git_commit_messages

[13] Jaechang Nam, Song Wang, Yuan Xi, Lin Tan, (2018). Designing bug detection rules for fewer false alarms, Proceeding ICSE '18 Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, 315-316

강화학습을 적용한 주행모드 변화에 따른 하이브리드 자동차 배터리 SOC 밸런싱 연구

김영균, Sangeeta Dey, 이석원
아주대학교
hotcrayz@ajou.ac.kr, deysangeeta@ajou.ac.kr, leesw@ajou.ac.kr

Study on Hybrid Electric Vehicle Battery SOC Balancing with Driving Mode changing using Reinforcement Learning

Kim Young Kyun, Sangeeta Dey, Lee Seok Won
Ajou University

요 약

본 연구는 강화학습을 적용한 하이브리드 자동차의 효율적에너지 분배에 관한 연구이다. 하이브리드 자동차의 에너지효율 개선과 관련해 많은 연구들이 현재 진행되고 있다. 에너지 효율 개선을 위해 동적 계획법, 규칙 기반 학습 등 다양한 방법들로 문제해결을 시도하고 있다. 에너지 분배와 관련된 여러 방법들 중 본 연구는 배터리 SOC(State Of Charge)밸런싱 과정으로 그 문제해결을 시도한다. 배터리 SOC 밸런싱 과정은 강화학습의 한 종류인 Q-Learning 으로 진행하였다. Q-Learning 적용 후, 변화한 배터리 SOC 값으로 세가지 주행모드에서 기존과 에너지 효율, 에너지 사용, 연비 등 여러 측면에서 비교과정을 진행한다.

1. 서 론

자동차가 발명된 이래로 효율적인 에너지 분배는 중요한 연구대상이다. 이를 위해 엔진과 관련된 여러 형태의 자동차가 등장했다. 본 연구에서 다루는 하이브리드 자동차 또한 효율적인 에너지분배의 목적으로 개발되었다.

하이브리드 자동차 성능개선과 관련해 많은 연구들이 진행되고 있다. 선행연구[1] 에서는 내연기관엔진과 전기모터엔진 사용의 최대 효율을 찾기 위한 방법으로 규칙기반 학습과, 강화학습의 한 종류인 TD-Learning 의 두 가지 방법을 적용한다. 여러 지표들로 두 가지 학습방법을 비교한 뒤 강화학습의 성능이 우수하다는 것을 설명한다. 선행연구[2]는 강화학습에 신경망구조를 추가하여 진행한다. 규모가 크고 변수가 많은 문제에서는 신경망 구조를 통해 효과적으로 다룰 수 있으므로, 해당 연구에서는 이를 통해 문제해결을 시도한다. 선행연구[3]도 강화학습과 신경망구조를 결합한 DQN (Deep Q-Learning Network)로 에너지 사용 성능개선을 연구한다. 선행연구[3]에서는 현재 t 시간의 보상(Reward)값으로 t+1 의 에너지사용 Action 을 결정한다. 내연기관엔진, 전기엔진 사용 두 가지 중 하나가 Action 으로 정의되는데, 본 연구에서는 배터리 SOC 에 초점을 두어 배터리 SOC 값의 이전, 다음 상태의 전이를 Action 으로 정의하였다.

본 연구는 하이브리드 자동차 에너지상 배터리 SOC 값을 사용해 하이브리드 자동차 에너지 효율 향상을 시도한다. 강화학습의 한 종류인 Q-Learning 을 사용한 후, 세가지 주행모드에서 기존 방법과 에너지효율, 에너지 사용, 연비 등을 비교한다.

2. 본 론

2.1 강화학습 모델

하이브리드 자동차의 에너지 효율을 높이기 위해 기존 연구에서는 동적계획법, 규칙 기반 학습 등 각각의 문제 목적에 맞는 다양한 방법들을 제시하였다. 본 연구에서는 에너지 효율을 높이기 위한 방법으로 강화학습을 적용하였다. 이는 최대의 보상을 받는

쪽으로 다음행동을 결정하는 강화학습이 본 연구와 취지가 맞아 위와 같은 결정을 했다.

본 연구에서는 강화학습의 Q-Learning을 적용하였다. Q-Learning은 강화학습 중 TD(Time Difference)-Learning의 한 형태이다. TD-Learning에는 몇 가지 형태가 존재하는데, Q값을 갱신하는 부분에서(그림 1의 Update 부분) 수정된 Q값의 Max 값을 곱하는 알고리즘의 차이를 보인다.

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q
    Take action a, observe r, s'
    Update
      $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s';$
  Until s is terminal

[그림 1] Q-Learning[8]

2.2 데이터

본 연구는 MATLAB 기반 자동차 시뮬레이션 프로그램인 ADVISOR 를 사용했다[7]. ADVISOR 프로그램은 자동차를 구성하는 여러 요소들을(전기모터, 동력계통, 바퀴 등등) 사용자 요구에 맞게 변경할 수 있다. 따라서 사용자 취향에 맞게 조절할 수 있어 본 프로그램을 사용하였다

ADVISOR 프로그램을 사용해 진행한 자동차 모델은 '2003 Toyota Prius Hybrid Car' 이다. 연구의 목적이 가솔린 엔진, 전기 엔진 두 개의 효율적 에너지분배 이므로 하이브리드 자동차로 진행했다.

자동차를 실험할 조건인 주행 환경도 중요하다. 에너지사용이 주행환경에 따라서 변화하므로 본 연구에서는 이를 고려했다. 연구는 ADVISOR 에서 제공하는 세 가지의 주행모드 환경에서 진행했다. 주행모드는 다음과 같다.

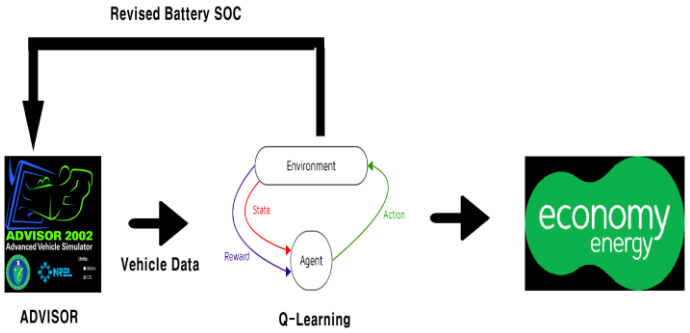
NYCC(New York City Cycle): 599 초

UDDS(Urban Dynamometer Driving Schedule):137 초

WVUCITY(West Virginia University): 1409 초

강화학습을 적용하기 위해서는 상황에 맞는 State 와 Action 의 사전정의가 필요하다. 본 연구는 배터리 SOC 값으로 하이브리드 자동차의 효율적 에너지분배에 관한 문제이다. 연구에서 State 는 시간 t 에서 배터리 SOC 값, Action 은 이전, 다음상태 두가지 중 한가지의 전이로 진행했다.

2.3 강화학습 프레임워크

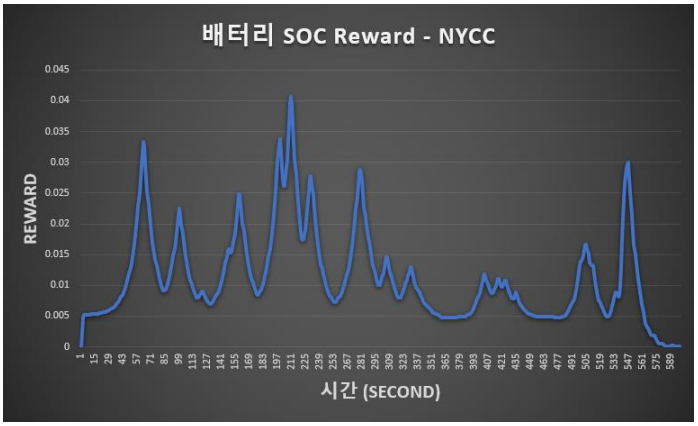


[그림 2] 본 연구의 프레임워크[6][7]

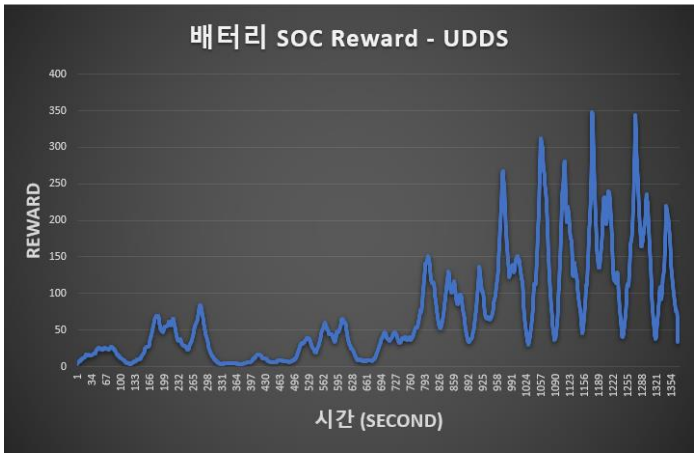
연구의 전반적인 강화학습 프레임워크는 [그림 2]와 같다. ADVISOR 로 수집된 데이터는 Q-Learning 의 입력 값이 된다. Q-Learning 학습 후 계산된 목표 배터리 SOC 값은 ADVISOR 로 재입력 값이 되어 다시 Q-Learning 으로 학습된다. 본 연구는 이와 같은 프레임워크를 통해 기존의 방법과 강화학습을 적용한 방법 두가지의 에너지 효율을 비교한다.

ADVISOR 로 정의한 하이브리드 자동차는 모델별로 목표 배터리 SOC 값을 갖는다. 이 값은 강화학습 적용 후 주행 모드에 따라 달라진다. 본 연구에서는 각각의 주행 모드에 따른 최적의 목표 배터리 SOC 값을 찾아 비교한다. 최적의 목표 배터리 SOC 값을 찾는 과정은 Q-Learning 의 결과 값인 Reward 로 계산한다. Reward 에 관한 내용들은 'Section-3. 결과'에서 설명한다

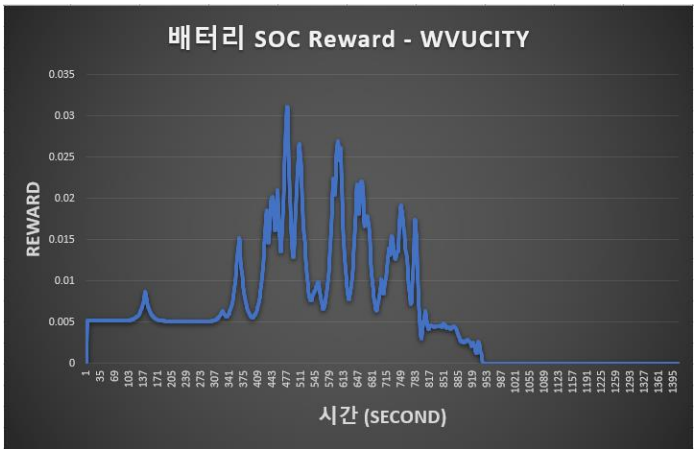
3. 결 과



[그림 3] 배터리 SOC의 Reward 값 변화-NYCC 모드



[그림 4] 배터리 SOC의 Reward 값 변화-UDDS 모드



[그림 5] 배터리 SOC의 Reward 값 변화-WVUCITY 모드

[그림 3], [그림 4], [그림 5]는 각각 주행모드의 시간 흐름에 따른 Reward 값 변화 그래프이다. Q-Learning 적용 후, Reward 값이 가장 높은 State의 배터리 SOC 값을 목표 배터리 SOC으로 설정했다. 강화학습에서는 학습 결과인 Reward를 최대로 하는 것을 목표로 한다. Reward를 더 많이 받는 쪽으로 Action이 결정되므로, 본 연구에서는 배터리 SOC 값이 가장 높은 값으로 결정된다. Reward 값 계산식은 [그림 6]이고[1], 강화학습 적용 후 주행모드 별 변화한 목표 배터리 SOC 값은 [표 1]과 같다.

Reward : $\dot{m}_f^{opt} \cdot \Delta T$, m: 시간 t에서 연료소비량, delta T : time step

[그림 6] Reward 계산 식

주행 모드	기존 목표 배터리 SOC	강화학습 후 목표 배터리 SOC
NYCC	60%	54%
UDDS	60%	49%
WVUCITY	60%	48%

[표 1] 목표 배터리 SOC 값 변화

[표 1]은 세가지 주행모드에서 기존과 강화학습 적용 후 목표 배터리 SOC 값 변화를 나타낸다. NYCC, UDDS, WVUCITY에서 변화한 목표 배터리 SOC 값은 차례로 54%, 49%, 48% 이다. 세가지 주행모드에서 공통적으로 강화학습 후의 목표 배터리 SOC 값이 감소하는 결과를 보였다.

강화학습 후의 목표 배터리 SOC 값들은 위에서도 설명했듯, Reward 값이 가장 높은 상태일 때의 값으로 결정했다. 이는 [그림 3], [그림 4], [그림 5]에서 Reward 값이 가장 높은 상태일 때의 배터리 SOC 값에 해당한다.

	기존 (목표 배터리 SOC= 60%)			강화학습 후 (목표 배터리 SOC= 54%)		
발전기 효율	In_energy(kj)	329.3	90.13%	In_energy(kj)	517.4	92.64%
	Out_energy(kj)	296.8		Out_energy(kj)	479.3	
전기모터 효율	In_energy(kj)	878.3	74.73%	In_energy(kj)	951.5	74.72%
	Out_energy(kj)	656.4		Out_energy(kj)	710.9	

[표 2] 에너지 효율-NYCC 모드

	기존 (목표 배터리 SOC= 60%)			강화학습 후 (목표 배터리 SOC= 49%)		
발전기 효율	In_energy(kj)	661.5	69.09%	In_energy(kj)	801.0	93.26%
	Out_energy(kj)	457.0		Out_energy(kj)	747.1	
전기모터 효율	In_energy(kj)	2470.0	80.69%	In_energy(kj)	2246.6	80.88%
	Out_energy(kj)	1998.7		Out_energy(kj)	1817.1	

[표 3] 에너지 효율-UDDS 모드

	기존 (목표 배터리 SOC= 60%)			강화학습 후 (목표 배터리 SOC= 48%)		
발전기 효율	In_energy(kj)	492.6	66.22%	In_energy(kj)	890.2	97.60%
	Out_energy(kj)	326.2		Out_energy(kj)	868.8	
전기모터 효율	In_energy(kj)	1151.6	80.27%	In_energy(kj)	969.5	77.73%
	Out_energy(kj)	924.4		Out_energy(kj)	753.6	

[표 4] 에너지 효율-WVUCITY 모드

[표 2], [표 3], [표 4]는 각각의 주행모드에서 에너지 효율 변화를 보여준다. 세개의 주행모드 각각 발전기의 효율이 90.13% → 92.64%, 69.09% → 93.26%, 66.22% → 97.60%로 증가했다. 특히, UDDS, WVUCITY 주행모드에 그 증가 폭이 20% 넘게 된 것을 볼 수 있다. 그에 비해 전기모터효율은 각각 74.73% →

74.72%, 80.69% → 80.88%, 80.27% → 77.73%로 강화학습 후의 결과가 오히려 감소하는 경우도 있었다.

전체 사용량 측면에서 살펴보면, NYCC 주행모드에서 주입된 에너지는 329.3kj + 878.3kj =1207.6kj에서 517.4kj + 951.5kj = 1468.9kj 로 약 260kj 정도 증가했다. 발전기의 효율은 개선되었지만, 전체 에너지 사용량은 오히려 더 많아졌다. 이는 같은 주행모드에서 더 많은 에너지가 소모되었다는 의미이다. 따라서 에너지 효율 상승은 유지하면서, 전체 에너지 사용량에 대한 추가적인 개선이 필요하다는 것을 알 수 있다.

[표 3]의 UDDS, WVUCITY 주행모드에서는 발전기 효율이 69.09% → 93.26%, 66.22% → 97.60%로 NYCC 주행모드와 비교 시 크게 향상되었다. 그러나 전기모터 측면에서 보면, UDDS 주행모드에서 주입된 전기 에너지량은 2470kj → 2246.6kj로, WVUCITY 주행모드에서는 1151.6 → 969.5kj로 각각 감소하였다. 일반적으로 전기차의 연료비는 휘발유, 경유차와 비교 시 1/10 ~ 1/5 정도이다[9]. 하이브리드 자동차에서 전기모터를 더 많이 사용하는 것이 연비 측면에서 효율적이라는 의미이다. UDDS, WVUCITY 주행모드에서는 강화학습 후의 결과가 효율 측면에서는 개선되었지만, 연비 측면에서는 오히려 감소하는 결과를 보였다.

4. 결 론

본 연구는 강화학습(Q-Learning)을 적용해 세가지 주행모드에서 하이브리드 자동차의 효율적 에너지사용 연구를 진행했다. 효율적 에너지사용 분석을 위해 배터리 SOC 값을 사용했고, 이를 Q-Learning 에 적용했다. State, Action 변화에 따라 보상 값이 변하는 Q-Learning 을 적용해 발전기 에너지효율이 개선된 것을 확인했다. 그러나 전기모터에서는 미세한 효율변화를 보였다.

연비측면에서 보면, 앞에서 설명했듯 UDDS, WVUCITY 주행모드에서 전기모터 사용량이 감소해 더 많은 비용이 소모되었다. 연구에서는 발전기, 전기모터 엔진사용만 다루고 연비를 고려하지 않은 것으로, 비용적 측면까지 고려한 추후 연구가 필요하다.

또한, 본 연구는 배터리 SOC 값 지표 하나만 가지고 실험을 진행했으나, 하이브리드 자동차 에너지 사용에는 상당히 많은 요소들이 영향을 끼친다. 이와 관련된 토르크, 연료 배출량 등의 다양한 지표들을 추가해 추후 연구를 진행할 것이다.

이 논문은 현재 데이터 분석에 집중된 결과만을 보여주고 있지만 추후에 자동차 소프트웨어와 하드웨어 콘트롤 시스템, 소프트웨어 디자인 및 성능분석에도 응용되어 쓰일 수 있고, 지능 소프트웨어 시스템 공학에도 필요한 연구라 생각된다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW 중심대학사업의 연구결과로 수행되었음(2015-0-00908)

참 고 문 헌

[1] Xue Lin, Yanzhi, Paul Bogdan, Naehyuck Chang, and Pedram, “Reinforcement Learning Based Power Management for Hybrid Electric Vechicles”, 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)

[2] Yue Hy, Weimin Li, Kun Xu, Taimoor Zhid, Feiyan Qin, and Chenming Li, “Energy Management Strategy for a Hybrid Electric Vehicle Based on Deep Reinforcement Learning”, Applied Sciences (2076-3417) Feb2018, Vol. 8 Issue 2, p1-N.PAG. 15p

[3] Xuewei Qi, Yadan Luo, Guoyuan Wu, Matthew J.Barth, “Deep reinforcement learning-based vehicle energy efficiency autonomous learning system”, 2017 IEEE Intelligent Vehicles Symposium (IV)

[4] T. Markel, A. Brooker, T.Hendricks, V.Johnson, K.Kelly B.Kramer, M.O’Keefe, S.Sprik, K.Wipke, “ADVISOR: a systems analysis tool for advanced vehicle modeling”, Volume 110, Issue 2, 22 August 2002, Pages 255-266

[5] Sunyoung Kim, Hyuk Lim, “Reinforcement Learning Based Energy Management Algorithm for Smart Energy Buildings”, Energies, Volume 11, Issue 8(August 2018)

[6] My Journey Into Deep Q-Learning with Kears and Gym, “Q-Learning”, https://medium.com/@gtnjuvin/my-journey-into-deep-q-learning-with-keras-and-gym-3e779cc12762, (2018.12.28)

[7] ADVISOR Advanced Vehicle Simulator, “ADVISOR”, http://adv-vehicle-sim.sourceforge.net/, (2018.12.28)

[8] ResearchGate, “Q-Learning algorithm”, https://www.researchgate.net/figure/The-Q-Learning-Algorithm-6_fig1_220776448, (2018.12.28)

[9] 환경부 전기차 충전소,”전기차”, https://www.ev.or.kr/portal/costCompare?pMENUMST_ID=21629, (2018.12.26)

수면 데이터 분석 플랫폼 설계 및 구현

전재민, 김성현, 서영균

경북대학교 컴퓨터학부

{jaeminjj,kshy9598,yksuh}@knu.ac.kr

Design and Implementation of Sleep Data Analytics Platform

Jaemin Jeon, Seounghyun Kim, Young-Kyoon Suh

School of Computer Science and Engineering, Kyungpook National University

요 약

최근 수면 데이터와 개인의 건강의 상관 관계 대한 연구가 활발하게 진행이 되고 있다. 본 논문은 10ms 마다 생성되는 수면 데이터에 대한 확장성 있고, 개인화된 분석 서비스를 제공하기 위해 설계된 플랫폼인 SCRAP (SleepCaRekit data Analytics Platform)을 소개한다. 제안된 플랫폼은 실시간으로 수집되는 수면 데이터를 효율적으로 저장하고 분석할 수 있는 유연한 소프트웨어 스택을 갖는다. 현재, SCRAP 시스템은 하루 당 약 18M 튜플로 구성된 수면 데이터를 빠르고 효과적으로 분석하여 개인의 건강 상태와의 상관 관계를 알려줄 수 있다. 추가적으로, 수면 분석 데이터에 대한 프라이버시를 보장해 줄 수 있도록 블록체인 적용을 고려한다. 끝으로, SCRAP 시스템을 활용한 수면 분석 결과를 보인다.

1. 서 론

'수면 중 무호흡증'[1]은 현대인들에게 최근 많이 발생하고 있는 수면 장애 중 하나이다. 특히, 삶의 질에 직접적인 악영향을 미치기 때문에 중요한 수면 장애로 여겨진다. 더욱이, 과도한 무호흡증으로 정신 지체, 운동 능력 저하 및 성격장애 같은 현상이 일어 날 수도 있는 것으로 보고되고 있다 [2].

이러한 무호흡증의 원인을 분석하기 위해, 기존 병원은 고가의 장비를 사용하여 수면 데이터를 추출해 왔다. 수면 데이터 연구에서 많이 쓰이던 야간 다발도 검사(night-time polysomnography PSG)는 장기 모니터링을 위한 비용과 시간 문제를 안고 있다. 다행히, 최근에 ECG 패치 및 PPG 기반 스마트 시계 같은 착용식 장치의 사용 [3,4]의 증가로 휴대성이 강화되었다. 특히, PAAR TAG Pillow와 Homecarekit를 사용한 SleepCareKit[5]은 웨어러블 기기로 수면 상태를 실시간으로 추정 및 분석할 수 있다. 일정 상황 발생 시, 사용자에게 신호를 보내 사용자가 즉각적인 피드백을 받을 수 있게 해 준다.

그러나, SleepCareKit과 같은 휴대용 수면 측정 기기들은 방대한 양의 데이터를 저장하고 분석하는데 본연의 한계가 있다. SleepCareKit의 경우 10ms당 텍스트 기반의 원시 데이터의 수면 데이터 한 열이 생성이 된다. 구체적으로, 사람 당 한 시간에 36만줄씩, 하루에 평균 5시간의 수면을 취한다고 하면 대략 180만줄의 데이터가 수면 데이터로 생성이 된다. 이러한 데이터를 제한된 용량의 휴대용 기기 내에서 저장, 분석 하기 어렵다. 만약, 이러한 수면 데이터를 효율적으로 저장하고 분석할 수 있는 플랫폼이 있다면 위에서 언급된 수면 중 무호흡증의 요인들에 대한 상관 관계 분석이 용이해 지고, 더 다양한 수면 데이터 분석이 이루어 질 수 있을 것이다.

본 논문은 대량의 수면 데이터를 수집, 분석하여 사용자에게 개인화된 수면 서비스를 제공해 줄 수 있는 플랫폼인 SCRAP (SleepCaRekit data Analytics Platform)을 소개한다. 구체적으로, SCRAP 시스템은 방대한 양의 수면 데이터를 분산 저장할 수 있으며, 프라이버시를 보장하기 위해 블록체인 기술을 통해 보안성을 강화를 고려한다. SCRAP은 기존 연구와는 달리, 산소 포화도 (SPO2), 호흡량 (Breathe Volume), 가속도 (Accelerator data), 심장 박동수 (Heartrate) 등의 다양한 (SleepCareKit) 수면 데이터를 활용해 무호흡증과의 상관 관계 분석을 수행할 수 있다. 본 논문에서 우리는 SCRAP 시스템의 소프트웨어 아키텍쳐 및 실제 관계 분석 결과를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 제시하고, 3장에서 SCRAP 시스템을 제안한다. 4장은

SCRAP에 축적된 SleepCareKit 수면 데이터에 대해, 가속도 센서 및 수면 자세와의 분석 결과를 보여준다. 마지막으로, 5장에서 본 논문의 결론을 맺고 추후 연구 진행 방향에 대해 논의한다.

2. 관련 연구

기존 연구 [6]은 Derby 데이터베이스와 자바 서버를 활용하여 환자의 수면데이터를 가져와서 무호흡증을 측정한다. 분석은 자바 서버에서 이루어지고 데이터베이스에 저장이 되고 웹사이트를 통해서 사용자가 정보를 받을 수 있다. 환자는 자신의 데이터를 쉽게 전달받을 수 있다. 또 다른 연구[7]는 산소 포화도 센서를 통해 무호흡을 측정한다. 원격 환자에게 무호흡과 산소 포화도를 측정한 결과를 게이트웨이를 통하여 클라우드 기반 시스템 아키텍처에 신호를 보낸다. 대량의 수면 데이터를 활용하기 위해서 아마존 클라우드를 사용해서 확장성을 보장한다. 클라우드 사용으로 분석, 저장 및 시각화를 강화한다.

하지만 매일 축적되는 데이터를 분석하고 보관하는 확장성 있는 시스템이 필요하다. 또한 단지 데이터에 대한 정보뿐만 아니라 그래프화를 통해서 환자들을 위한 시각화가 부족하다.

본 논문에서 제시하는 SCRAP은 오픈소스인 아파치 스파크 (Apache Spark)[8]를 사용해서 수면 데이터로부터 추출된 다양한 값들을 함수형 프로그래밍을 사용해서 상관관계를 분석하여 그래프화를 시켜서 환자들이 쉽게 접근할 수 있도록 돕는다.

더욱이, 위에서 언급된 기존 연구들은 게이트웨이 사용으로 이해 보안성 문제를 인식하지 않았으며 대량의 데이터를 보관 및 분석하는데 문제점이 존재한다. 이런 문제점을 해결하기 위해 SCRAP은 블록체인 기술을 활용한 보안성 강화를 고려한다. 그리고 도커/컨테이너 (Docker/Container) 기술을 활용한 개인화된 컨테이너를 구현하여 개인화된 수면 데이터 분석 서비스를 제공할 수 있게 설계되었다.

3. SCRAP 시스템 설계

SCRAP은 대용량의 텍스트화된 수면 데이터를 암호화를 통해 데이터베이스에 로딩과 HDFS와 Spark기술을 활용해 확장성 있게 적재할 수 있도록 하며 바로 분석 할 수 있도록 유연한 구조로 되어있다. SCRAP 시스템은 크게 저장 및 로딩 하는 과정과 분석 및 공급하는 과정으로 구성되어 있다.

본 논문은 그림 1과 같이 Sleepcarekit로부터 수집한 텍스트 기반의 대량의 원시 데이터를 효율적이며 저장 및 분석이 용이 하도록 돕는 플랫폼에 블록체인 기술을 활용하여 보안성을 강화하는 방안을 제안한다.

3.1 저장 및 적재

SleepCareKit를 통해서 수면 데이터를 텍스트 형식으로 받는다. 이 데이터들이 누적이 되어서 다양한 방법으로 활용을 하기 위하여 데이터베이스화를 한다. 우선 JDBC를 통해서 MYSQL에 넣는다. 넣는 과정에서 개인의 이름과 건강 정보, 아이디를 블록체인 관련 기술을 사용하여 보안성을 강화한다. 그리고 나서 Sqoop[9]을 통해서 하둡 분산 파일 시스템 (HDFS: Hadoop Distributed File System)[10]에 데이터 적재를 실시하며 이 하둡에 저장된 데이터를 도커/컨테이너 환경에서 관리되는 컨테이너를 사용하여 개인화된 분석을 지원하도록 설계되었다.

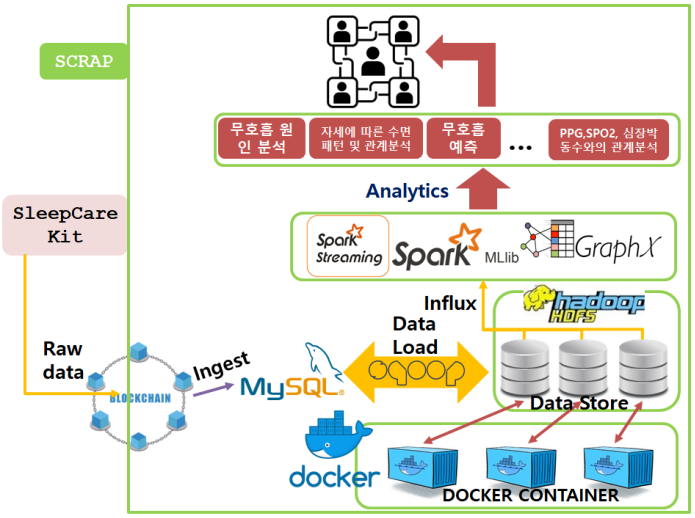


그림 1. SCRAP 시스템 구조

한 사람의 데이터는 한시간에 약 36만개의 tuple을 가지며 5시간의 수면시간으로 가정하면 하루에 180개의 튜플을 가진다. 개인의 호흡량 산소 포화도, 심장박동수 등의 대량의 데이터를 효과적으로 처리하기 위해서, 앞서 언급된 바와 같이 텍스트 기반의 원시 데이터를 JDBC를 통하여 MYSQL에 임포트를 하고 대용량 분산 분석을 위해 HDFS에 적재한다. 그렇게 되면 대량의 데이터를 스파크에 연동되는 다양한 라이브러리를 통해 무호흡이 측정되는 시점과 다른 수면 데이터와의 상관 관계 분석을 수행한다. SparkR[16]을 사용하여 무호흡이 일어나는 시점을 예측하여 환자에게 제공할 수 있게 한다.

데이터 적재의 경우 블록체인 기술[11]을 사용하여 개인의 아이디를 암호화를 통해서 저장을 하도록 한다. 본 연구는 모든 데이터의 암호화 보다는 데이터 분석결과에 대한 데이터만을 블록 체인화가 목적이다. 후에 스파크에서 분석한 데이터를 개인만이 볼 수 있도록 보안성을 보장한다. 예전에는 오직 탈중심화를 위해서 연구되었으나 최근에 블록체인 기술이 분산성과 보안성을 추구하기에 개인의 프라이버시를 강화한다.

3.2 관리 및 공급

도커/컨테이너 기술을 활용하여 개인화된 수면

데이터 분석 서비스와 현상 완화 가이드를 제공한다. 구체적으로, 컨테이너에 HDFS 및 스파크를 탑재하여 데이터를 확장성 있게 적재할 수 있하고 바로 분석할 수 있는 환경을 구성한다. 이 분석 데이터를 개인에게 바로 전달한다. 기존의 데이터베이스에서 대용량의 데이터를 처리하는데 시간적 문제가 많았으나 하둡을 사용함으로 방대한 데이터를 분산처리해서 효율적으로 처리할 수 있도록 하였다.

4. 수면 데이터 분석

SCRAP 시스템 설계 및 구축의 진행과 동시에 우선적으로 수면 데이터를 분석하였다. 무호흡 발생은 수면 자세와 관련이 있는 것으로 알려져 있다[12,13,14]. 수면 데이터 측정의 과정에서 이러한 점을 고려하여, 가속도 센서를 통하여 자세를 측정하고자 했다. 그러나, 가속도 센서의 값이 어떠한 자세를 나타내는지, 수면 중 사람들이 어떤 자세들을 취하는지에 대한 분석은 추후에 진행될 예정이다. 무호흡에 관한 분석에 앞서, 가속도 센서 데이터를 활용한 자세 분석을 진행하였다.

자세 데이터 분석 환경은 표 1과 같다.

표 1 자세 데이터 분석 시스템 환경

	환경
OS	Redhat 7.4
CPU	Intel Xeon Gold 5118(12core)
RAM	12GB
SSD	Crucial MX500 250GB M.2 Type
Language	R[15]
Interface	SparkR[16]

수면 중 취하는 자세의 가짓수와 그에 대한 가속도 센서 값을 확인하기 위하여 클러스터링을 진행하였다. 클러스터링의 군집 수를 결정하기 위하여 NbClust[17]를 사용하였다. 총 20명의 수면데이터를 기반으로 가속도 센서 값만을 입력 값으로 사용하였으며, 그 결과를 가시화한 것이 그림 2 이다. 그림 2로부터 알 수 있는 점은, 군집 수가 5일 때 가장 좋은 클러스터링 결과를 나타낸다는 점이며, 이를 바탕으로 5개의 군집으로 나눠지도록 클러스터링을 진행하였다.

클러스터링 알고리즘은 R언어의 `class` 패키지의 pam함수[18]를 사용하며, 군집 수는 5로 지정한다. 클러스터링 진행 결과를 가시화한 것이 그림 3이며, 각 군집의 center값은 표 1과 같이 나타난다. 표 1의 값을 살펴보면, 각 군집들의 가속도 센서 값의 차이가 확연히

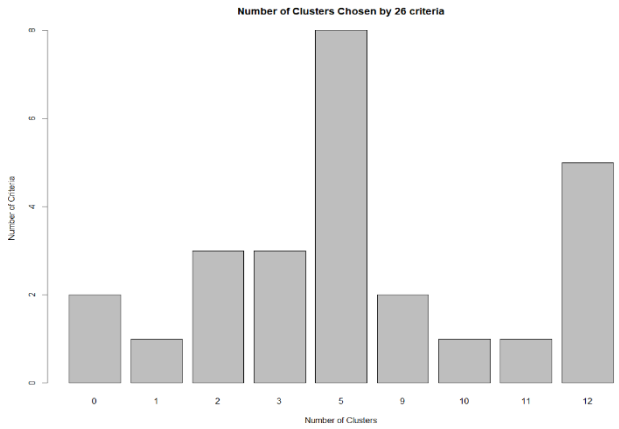


그림 2. 가속도 센서 데이터에 대한 자세 클러스터링

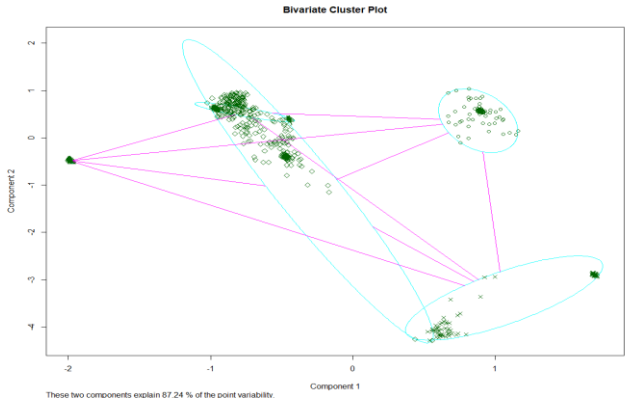


그림 3 :자세클러스터링 by PAM : 군집 수(K) = 5

표 2 각 군집의 액츄에이터 데이터

	Acc_x	Acc_y	Acc_z
1	224	251	201
2	20	0	192
3	3	207	212
4	241	206	37
5	238	3	195

눈에 보이며, 각 군집들 간의 자세 차이가 명확하다는 것을 의미한다.

클러스터링 결과를 바탕으로, 추후 수면 데이터 수집과 동시에 수면 자세를 직접 관찰한다면, 각 군집이 어떤 자세를 나타내는지 확인이 가능하다. 이러한 자세 분석 데이터는 추후 무호흡에 영향을 주는 요인으로 사용될 수 있으며, 결과적으로 무호흡을 개선하는 연구에 큰 도움이 될 것이라 기대한다.

5. 결론 및 향후 연구

본 논문은 대량의 수면데이터 분석을 통해 개인의 수면 패턴을 도출해서 무호흡증 분석을 통한 현상 완화 가이드를 개인에게 프라이버시와 보안성을 보장하며 줄 수 있는 수면 데이터 분석 플랫폼 (SCRAP)을

제안하였다. 기존의 수면 데이터 플랫폼과는 달리 방대하고 다양한 데이터를 분산처리 시스템을 통해서 효율적으로 관리해줄 수 있을 것으로 기대된다. 하루의 수면 데이터 뿐 만 아니라 한 사람의 누적된 데이터를 가지고 다양한 분석을 해 볼 수 있을 것이라고 예상된다. 도커/컨테이너를통해서 개인화된 수면 데이터 분석 서비스를 제공한다.

향후에는 현재 구현한 자세와의 무호흡 상관 관계 분석 말고도 다양한 분석을 할 예정이다. 우선 산소포화도, 심장박동 수, PPG 데이터와의 상관관계를 분석할 것이다. 그리고 무호흡이 일어나기 전의 데이터를 분석하여 무호흡이 일어나기 전에 어떤 증상이 일어나는지 언제 일어나는지를 예측 할 수 있도록 하여 환자들에게 그에 대한 피드백을 줄 수 있도록 하는 방안을 제시하도록 할 것이다.

사 사

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2018R1A6A1A03025109). 또한 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 신진연구자사업임 (No. NRF-2018R1C1B6006409). 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2016-0-00145, 어떤 주제에 대한 빅데이터를 스마트 보고서로 요약하는 기술 개발).

6. 참고 문헌

[1] R. V. S. N. Y. Sainath, K. Pruthvisai, B. V. H. A. R. Akhil and S. Palaniswamy, "Sleep Pattern Monitoring and Analysis to Improve the Health and Quality of Life of People," *2018 International Conference on Advances in Computing, Communications and Informatics* (ICACCI), Bangalore, India, 2018, pp. 1900-1905.

[2] Young, T., Peppard, P.E., and Gottlieb, D.J., "Epidemiology of obstructive sleep apnea: a population health perspective," *Amer. J. Respiratory Crit. Care Med.,* 165 (2002), pp. 1217-1239.

[3] P. Lévy, M. Kohler, W. T. McNicholas, F. Barbé, R. D. McEvoy, V. K. Somers, L. Lavie, J.-L. Pepin, "Obstructive sleep apnea syndrome," *Nature Reviews Disease Primers*, vol. 1, pp. 15015, 2015.

[4] T. Penzel, "Technology to assess sleep," *Sleep Medicine Clinics*, vol. 11, no. 4, pp. i, 2016.

[5] G. Shein, S. Sathynaryanan, A. Sagahyroon and M. Pasquier, "Sensors equipped mattress for assessing sleep apnea symptoms," *IEEE Conference on Wireless Sensors* (ICWiSe), November 2017, pp. 1-5.

[6] Li Haoyu, Li Jianxing, N. Arunkumar, Ahmed Faeq Hussein, Mustafa Musa Jaber, "An IoMT cloud-based real time sleep apnea detection scheme by using the SpO2 estimation supported by heart rate variability," *Future Generation Computer Systems,* 2018,

[7] 전영준, 최문식, 강순주. "다중생체신호 실시간 분석을 통한 수면 품질 진단 및 위험상황제어를 위한 웨어러블 소프트웨어구조." *한국정보과학회 학술발표논문집,* (2017.12): 522-524.

[8] Spark, https://spark.apache.org/ (2018년 10월 확인)

[9] Apache Sqoop, http://sqoop.apache.org/ (2018년 12월 확인)

[10] Apache Hadoop, https://hadoop.apache.org/ (2018년 12월 확인)

[11] Trent McConghy, Rodolphe Marques, Andreas Muller Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto, "BigchainDB: A Scalable Blockchain Database," white paper, BigChainDB, 2016.

[12] E. J. Olson W. R. Moore T. I. Morgenthaler P. C. Gay B. A. Staats "Obstructive sleep apnea-hypopnea syndrome" *Mayo Clin.* Vol. 78, no. 12, pp. 1545-1552, 2003.

[13] Ravelo-García A., Saavedra Santana P., Juliá-Serdá G., Navarro-Mesa J., Navarro, Esteva J., Álvarez López X., et al., "Symbolic dynamics marker of heart rate variability combined with clinical variables enhance obstructive sleep apnea screening", Chaos, 24 (2014), p.024404.

[14] 이은상, 최지호, 이보라, 정재현, 선상우, 이승재 김효준, 이재용, "Effect of Positional therapy on Patients with Obstructive Sleep Apnea: Meta-Analysis," *순천향대학교 의과대학 임상시험센터* J Rhinol 2017:24(2): 94-103.

[15] R, https://www.r-project.org/ (2018 년 11 월 확인)

[16] Spark-R, https://spark.apache.org/docs/latest/sparkr.html (2018년 11월 확인)

[17] Malika Charrad, Nadia Ghazzali, Veronique Boiteau, Azam Niknafs, "NbClust: An R Package for Determining the relevant Number of Clusters in a Data Set," *Journal of Statistical Software*, Vol. 61, Issue 66, October 2014.

[18]CLUSTER, https://cran.r-project.org/web/packages/cluster/cluster.pdf (2018 년 11 월 확인).

LoRa 기반의 치매환자를 위한 안전관리시스템

정재호, 김홍준, 최민철, 이보경*

한국산업기술대학교

{vhok74, tkdlqm2, tmxk4197} @kpu.ac.kr, bklee@kpu.ac.kr*

LoRa-based Safety Management System for Alzheimer's Patient

Jaeho Jung, Hongjoon Kim, Mincheol Choi, Bo Kyung Lee*

KOREA POLYTECHNIC UNIVERSITY

요 약

치매노인 실종문제를 해결하기 위해 최근 사물인터넷 기반의 배회감지기 제품들이 많이 나오고 있지만 기존 제품들은 8시간마다 충전이 필요하고 이동통신망을 사용하기 때문에 통신비를 지불해야한다. 본 논문에서는 저전력 장거리 무선통신 기술인 LoRa 통신을 이용하여 자체적인 시스템을 구축하고 기존 제품들의 단점들을 보완하는 안전관리시스템을 제안한다. 치매노인에게서 전송되는 위치정보를 스마트폰의 구글지도를 통해 실시간으로 확인할 수 있을 뿐만 아니라 심박수와 운동량 측정 기능을 통해 건강을 주기적으로 점검할 수 있는 기능을 추가하였다. 해당 시스템은 GPS 센서, 심박 센서 그리고 가속도 센서를 이용해 부가적인 정보를 제공하도록 구현하다. 또한 본 논문에서 구현한 시스템을 이용하여 LoRa 망이 지원하는 데이터 커버리지를 측정하여 기존의 망 보다 넓은 커버리지를 지원하는 것을 분석하였다.

1. 서론

지난해 치매환자의 수가 70만명을 넘어선 것으로 추정된다. 고령화 시대로 접어들면서 노인들의 수는 점차 증가하고 노인 10명 중 1명이 치매환자로 12분당 1명씩 발생한다. 매년 100여건 이상의 치매 노인 실종 신고가 접수되고 실종 신고가 들어오면 많은 경찰인력이 투입되어 수색에 나선다.[1] 빠르면 한시간 이내로 발견되지만, 실종 후 일주일의 수색 끝에 발견이 되는 경우도 있고, 끝내 숨진 채로 발견되는 안타까운 경우도 많다. 이러한 문제를 해결하기 위해 치매노인 실종시에 신속하게 위치를 파악할 수 있는 IoT 기반 배회감지시스템이 개발되고 있다. 시중에 나와있는 배회감지기 제품들은 대부분 이동통신망을 이용하여 치매노인의 위치정보를 송수신 하기 때문에 넓은 커버리지를 가진다는 장점이 있다. 하지만 통신망 이용에 대한 이용요금이 발생하고 8시간마다 충전해야 한다는 불편함이 있으며, 자신이 치매라는 것을 인정하기 싫어 해당 배회감지기 제품을 착용하는 것을 거부하는 사람들도 있다. 따라서 전국 치매 환자 24만명 가운데 배회감지기 이용자는 3천6백명 정도로 1.3%에 불과하다.[2]

본 논문에서는 이러한 단점을 개선하기 위해 저전력으로 장시간 사용이 가능하고 장거리 통신이 가능한 LoRa 기반 배회감지 기능을 갖는 안전관리시스템을 제안한다. 해당 시스템은 LoRa 모듈 간의 무선통신 거리가 약 1.2km이고, 기존 이동통신사망을 사용하지 않고 자체적인 시스템을 사용하기 때문에 통신비가 들지 않는다. 본 시스템은 위치 정보뿐 만 아니라 심박수 측정 기능을 통해 환자의 상태를 주기적으로 확인 가능하고 일정 기간동안 움직인 횟수를 통해 운동량을 측정하여 환자의 건강 상태를 관리할 수 있다. 보호자는 이러한 정보들을 스마트폰으로 전달받아 실시간으로 치매환자의 상황을 알 수 있다. 2장에서는 기존 배회 감지시스템에 대하여 설명하고 3장에서는 본 논문에서 제안한 치매환자를 위한 안전관리시스템에 대한 소개와 제안한 시스템을 활용하여 데이터의 커버리지를 측정하고 분석한다. 마지막 4장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1 기존 배회감지시스템

17 년 8 월부터 경찰청은 SK 하이닉스와 배회감지기 무상보급 업무협약을 맺고, 상습 실종 치매노인 1 만명에게 무상보급을 시작하였다. 그림 1 은 SK 하이닉스에서 개발한 배회감지기 제품이다.

그림 1. SK 하이닉스의 배회감지기 제품

정부는 이를 통해 실종노인에 대한 정부의 적극적 지원을 확대했으며 배회감지기 도입으로 인한 치매노인 발견에 소요되는 시간이 크게 단축되었다.[3]

하지만 이러한 긍정적인 결과와는 반대로 배회감지기 보급률과 이용률은 매우 저조한 상황이다. 무상보급은 노인장기요양보험 등급을 받은 환자에게만 대여의 우선권이 주어지고 그렇지 못한 사람들에게는 이용에 대한 비용이 들게 된다. 또한 대여를 받기 위해서는 지원의뢰서, 배회감지기 계약서 사본, 통장계좌 사본 등의 첨부서류를 메일로 등록하여 신청해야 한다. 배회감지기 지원 기간은 1년으로 기기 대여비와 통신비가 지원되며 1년 후 부터는 기기 값과 통신비를 지불해야한다. 만약 지원기간 동안 기기를 분실할 시 약 20만원의 분실비용을 내야한다.[4]

정부의 무상보급 제품 이외에도 LoRa망을 사용하는 배회감지기 제품도 많이 개발되고 있다. SPACOSA사의 Gper, AT-매팟 등이 있다. 이 제품들은 기존 단순 구매비용과 통신비 이용요금만 지불하면 사용 가능하다. 무상 보급되는 제품들과는 달리 복잡한 사용절차가 없어 편리하다는 장점이 있다. 하지만 10만원 상당의 기기값과 장시간 사용을 위해 보조배터리를 별도로 구매할 경우 30만원이 넘어가는 비용이 발생한다. 또한 기기값과 별도로 매달 SK의 LoRa 통신망 사용에 대한 비용도 지불해야 한다.[5]

2.2 LoRa(Long Range)[6]

LoRa는 다양한 특징을 가지는 새로운 무선통신 기술이다. 저전력으로 장시간 사용이 가능하며 개발자 가이드에 따르면 10km이상 장거리 통신이 가능하다고 한다. 또한 다중 센서 기능을 가짐으로써, 노드에 여러 개의 센서를 부착해 사용이 가능하다.

최근 모든 사물을 인터넷에 연동시켜 사물들 간에 데이터를 전송하는 사물인터넷 기술이 다양한 분야에 적용되고 있다. 배회감지기도 그러한 사물인터넷 기반의 서비스 중 하나로 기존 블루투스나 지그비 같은 무선통신기술보다 통신범위가 넓고 장시간 경량 데이터를 전송할 수 있는 저전력의 장거리 무선통신 기술인 LoRa를 사용하면 더 나은 성능을 제공할 수 있다.

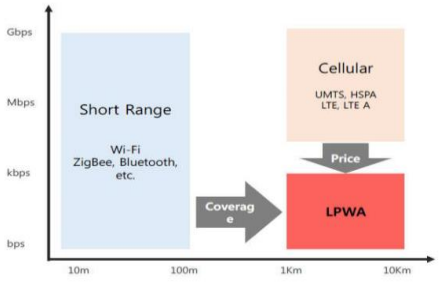


그림 2. LoRa 통신 특징

그림 2는 각 무선통신별 특징을 보여준다. 짧은 통신거리를 가지고 있는 와이파이, 지그비, 블루투스는 데이터 전송률이 큰 대신 짧은 커버리지를 가지고 있다. 셀룰러 통신은 높은 커버리지와 데이터 전송률을 보유하지만 이동통신망을 사용하는 것이기 때문에 높은 통신비가 발생한다. 이에 반해 LoRa 통신은 높은 커버리지와 저전력이라는 특징이 있기 때문에 위치 값이나, 센서 값 같은 경량데이터를 사용함에 있어 IoT에 최적화된 기술이다.

3. 구현시스템

3.1 시스템 구성도

본 논문에서 구현한 치매환자를 위한 안전관리시스템은 센서들로 구성된 감지시스템과 게이트웨이, 네트워크 서버 및 웹 서버로 구성된다. 치매환자로부터 수신되는 정보들이 게이트웨이와 네트워크 서버를 거쳐 웹 서버로 전송되고 최종적으로 클라이언트는 스마트폰으로 해당 정보를 확인할 수 있다. 감지시스템에는 심박 센서와 가속도 센서, GPS모듈이 동작되고 있으며 치매환자의 위치 값과 함께 현재 환자의 심박수와 활동량이 LoRa 통신을 통해 게이트웨이를 거쳐 네트워크 서버로 전송된다. 네트워크 서버에서는 게이트웨이로부터 전송되는 모든 데이터들을 관리하고 최종적으로 웹 서버를 통해 클라이언트의 스마트폰에 치매환자의 정보를 전송한다.

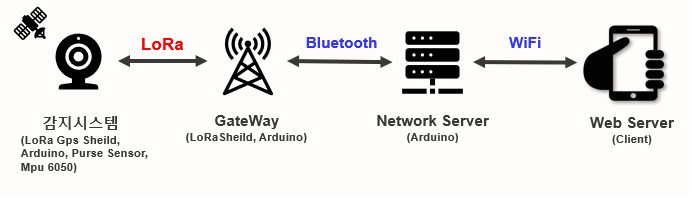


그림 3. 통합 시스템 구성

3.2 세부 기능

안전관리시스템은 감지시스템, 게이트웨이, 네트워크 서버, 클라이언트로 구성되며 구성 요소 별 세부 기능은 다음과 같다. 치매노인에 부착된 감지시스템에는 총 3가지의 센서 모듈이 필요하다. 우선 LoRa Shield에 부착된 GPS 모듈을 사용하여 위치 값을 수신 받는다. 심박 센서에는 가운데 LED에서 밝은 녹색의 빛이 나오며 바로 아래 반사되는 빛을 감지하는 센서가 있다. 심장박동시 혈류가 증가하게 되면 반사되는 빛의 양이 줄어든다는 점을 이용하여 심박수를 측정한다. 또한 MPU 6050 가속도 센서를 이용하여 운동량을 측정할 수 있다. 해당 센서는 6축 가속도/자이로 센서라고 불리며 가속도 3축, 자이로 2축, 온도 1축을 구성한다. 가속도 축과 자이로 축을 이용하여 센서의 흔들림 변화를 측정한다.

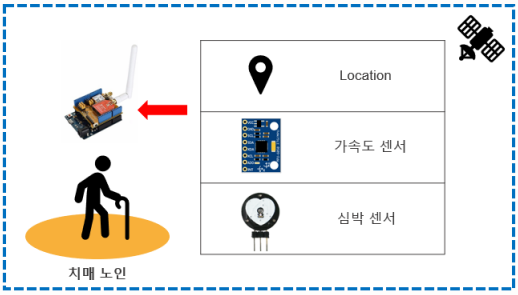


그림 4. 감지시스템 부착 센서

최종적으로 3가지의 기능(위치 값, 심박수, 흔들림 변화량)이 치매환자로부터 측정되면 해당 데이터는 LoRa 통신을 통해 게이트웨이로 전달된다. 게이트웨이는 LoRa Shield로 구성되어 있으며 감지시스템에서 전송되는 데이터를 수신 받는다. 위치 값(위도, 경도)값은 그대로 네트워크 서버로 전송을 한다. 심박수는 클라이언트 단에서의 오버헤드를 줄이기 위해 1분 동안 측정되는 심박수의 평균치를 계산하여 네트워크 서버로 전송한다. 마지막으로 가속도의 변화량인 AcX, AcY, AcZ의 값을 절대치로 합산하여 그 값이 800이상 출력될 경우 움직임이 한번 발생하였다고 판단하였고, 그 즉시 네트워크 서버로 활동량을 전송한다. 네트워크 서버는 여러 게이트웨이로부터 수신 받은 데이터를 최종적으로 종합하여 웹 서버를 통해 클라이언트에게 위치 값과 계산된 평균 심박수, 마지막으로 활동량을 송신한다. 클라이언트는 수신 받은 데이터를 스마트폰으로 확인하여 환자의 정보를 확인한다. 그림 5는 안전관리시스템의 시스템 제어 흐름도를 나타낸다.

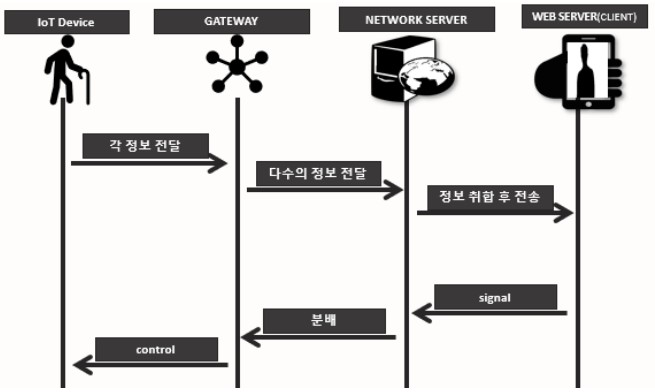


그림 5. 시스템 제어흐름도

클라이언트는 전달받은 위도와 경도 값을 이용하여 구글 맵을 통해 사용자의 위치를 실시간으로 확인할 수 있다. 만약 이동할 수 있는 안전 반경을 정하여 환자의 위치 값이 해당 반경을 벗어 날 경우 위험 알림을 통하여 빠른 사고 예방이 가능하다. 또한 1분 동안의 평균 심박수가 계속 수신되어, 심정지와 같은 사고를 예방할 수 있다. 또한 하루 목표치의 활동량을 설정하여 운동량에 대한 관리가 가능하다. 또한 네트워크 서버에서 종합된 정보가 웹 서버로 전송된다. 이에 따라 관리자는 스마트폰으로 환자의 정보를 확인한다. 관리자는 안전반경을 정해 놓고 환자의 위치 값이 해당 안전반경을 벗어날 경우 위험 알림이 울리도록 설정하였다.

3.3 시스템 구현

본 논문에서 구현한 안전관리시스템의 세부 구현 내용 및 모듈은 다음과 같다. 감지시스템은 각 모듈을 제어할 수 있는 Arduino uno r3에 LoRa 통신 및 GPS 수신을 할 수 있는 Dragino 사의 LoRa / GPS Shield 를 부착하였으며, 노인의 심장 박동수를 측정할 수 있는 심장 펄스 센서와 활동량을 측정할 수 있는 가속도 센서(MPU 6050)를 부착하여 그림 6과 같이 구성하였다. LoRa 통신과 GPS 수신을 위해 LMIC 와 TinyGPS 라이브러리를 사용하였으며, IDE 는 arduino-1.8.5를 사용하여 C 언어로 개발하였다. 게이트웨이는 각 모듈을 제어할 수 있는 Arduino uno r3에 LoRa 통신을 위한 Dragino 사의 LoRa Shield 를 부착하였으며, 네트워크 서버와 블루투스 통신을 위해 블루투스 모듈(HC-06)을 부착하여 그림 6과 같이 구성하였다. LoRa 통신을 위해 LMIC 라이브러리를 사용하였으며 arduino-1.8.5 개발환경을 이용하여 구현하였다. 네트워크 서버는 Arduino uno r3에 블루투스 모듈(HC-06)을 부착하여 그림 7과 같이 구성하였다.

그림 6. 게이트웨이(좌) 및 감지시스템(우)

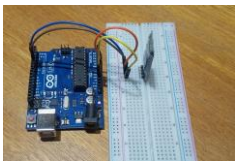


그림 7. Network Server

클라이언트는 Apache Tomcat을 이용하여 웹 서버를 구현하였으며 IP주소를 통해 8080포트를 허용하여 모바일 웹에서 확인할 수 있도록 하였다. IDE는 Eclipse를 사용하여 Java언어로 개발하였다. 웹 애플리케이션에서는 위치확인, 운동량, 심박수 탭을 만들어 각각의 기능을 확인할 수 있게 구현하였다. 위치확인 탭에서는 실제 최대 측정되는 거리를 구글 맵을 통해 확인한 결과이다. 운동량은 움직인 활동량을 측정한 결과이며 1분 동안의 평균 심박수를 확인할 수 있다. 또한 나이별 평균 분당

심박수를 통한 건강 상태를 지속적으로 주시할 수 있게 하였다. 그림 8은 클라이언트의 스마트폰에서 동작되는 웹 애플리케이션 화면이다.

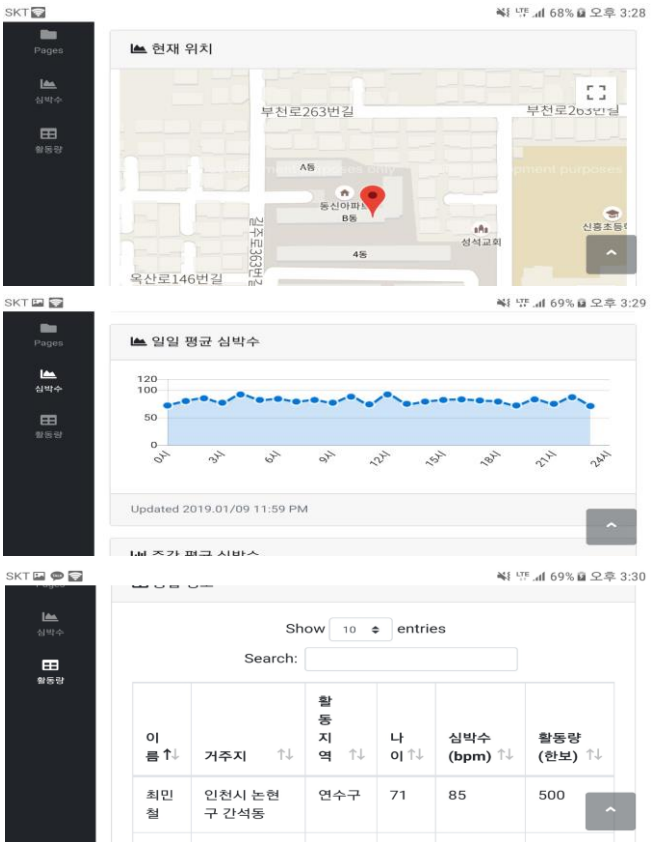


그림 8. 안전관리시스템 관리자 화면

3.4 LoRa 커버리지 측정

본 논문에서 제안하는 안전관리시스템에 적용된 장거리 통신 기술인 LoRa통신이 장거리에서 데이터 송수신이 가능한지 확인하기 위하여 거리에 따른 커버리지를 측정하였다. 성능 테스트에 사용된 LoRa모듈은 868 MHZ의 주파수 대역을 사용하며 최대 300 kbps의 데이터 전송률과 20dBM의 출력세기를 가진다.[7]

거리에 따른 LoRa모듈간의 데이터 송수신을 확인하기 위하여 게이트웨이 위치를 부천에 있는 부인초등학교로 선정하였으며 부인초등학교로부터 상동호수공원까지 이동하며 LoRa/GPS 수신 모듈의 데이터 송신 값을 확인하였다. 그 결과 게이트웨이에서 최대 1200m까지의 거리에서 데이터 송신이 가능하였으며 그 이상은 더 이상 송신이 이루어지지 않았다. 또한 GPS수신부와 게이트웨이 사이가 건물로 막혀 있으면 데이터 송신이 원활하지 못했다.

표1. 무선 통신 기술의 성능 비교[8-11]

		Wi-Fi(IEEE802.11n)	Zigbee	Bluetooth 5	LoRa
주파수(GHz)		2.4 / 5	0.868 / 0.915 / 2.4	2.4	0.434 / 0.868 / 0.915
속도(Mbps)		7.2 ~ 150	0.02 ~ 0.25	1 ~ 24	~ 0.01
전송거리(m)		70 ~ 250	30 ~ 100	40 ~ 400	10000 ~
전력소모(mW)		300	50	50	10 - 25
운영시간 (2000mAh AA기준)	연속사용	2-4시간	60시간	60시간	120시간
	대기	50시간	30일	30일	10년~20년
모듈 가격		$5-$12	$6-$12	$6-$12	$3

표1은 무선통신기술인 Wi-Fi, Zigbee, Bluetooth, LoRa의 성능을 비교한 것을 나타낸다. Wi-Fi는 현재 많이 사용하고 있는 IEEE802.11규격 중 802.11n이며 Bluetooth는 최신 버전이자 2017년부터 상용화되기 시작한 Bluetooth 5를 선택하여 비교하였다. LoRa는 다른 무선통신기술보다 데이터 전송속도는 낮지만 최대 10km이상 전송 가능하다. 또한 Wi-Fi, Zigbee, Bluetooth에 비해 소비전력이 현저히 낮기 때문에 장시간 사용이 가능하고 가격부담이 없다. 따라서 이러한 점을 종합적으로 고려했을 때, LoRa 통신기술이 본 시스템에 가장 적합하였다.

4. 결론

기존의 배회감지기 제품은 보급을 받기 위한 절차가 까다로운 점과 이동통신망을 사용함으로써 넓은 커버리지를 갖는다는 장점이 있지만 그에 따른 별도의 통신비 부담이 존재한다. 이러한 문제점을 해결하기 위하여 자체적으로 구현한 시스템을 이용하여 통신비 요금 부담을 해결하고 저전력 장거리 통신인 LoRa를 사용하여 배터리 전력 소모를 줄여 먼 거리에서도 노인의 위치와 건강 상태를 확인할 수 있도록 구현하였다[12].

본 논문에서 제안하는 노인 안전관리시스템을 통하여 기존의 배회감지기의 저조한 보급률과 이용률 개선 및 현재 대두되고 있는 치매노인 실종 상황을 IoT기술을 통하여 예방할 수 있는 효과를 기대할 수 있다. 그러나 본 논문에서 제안한 시스템은 IoT환경에 적합한 응용, 전송 계층의 프로토콜이 아닌 기존의 웹 환경에서 지원되는 프로토콜을 사용하기 때문에 성능이 떨어질 수 있다. 또한 IoT 환경에서 송수신 데이터의 손실 및 위, 변조의 가능성이 있기 때문에 보안 프로토콜을 적용할 필요가 있다. 따라서 IoT 환경에 적합한 프로토콜을 이용하고 데이터의 안전성을 보장하기 위한 보안 프로토콜을 적용한 안전관리시스템을 개발하는 것을 향후 연구과제로 제안한다.

5. 참고문헌

[1]http://news.chosun.com/site/data/html_dir/2018/02/17/2018021700093.html
[2]http://mn.kbs.co.kr/news/view.do?ncd=3447353
[3]http://m.police.go.kr/m/bbs/view.do?nttId=20602&bbsId=B0000011&menuNo=2600252&delCode=0

[4]http://www.alzza.or.kr/bbs/board.php?bo_table=61&wr_id=11
[5]http://www.gper.me/ko/
[6]최광호, 진성배, 이충완, 이민선, “LoRa를 이용한 무선 통신의 성능 측정 연구”,한국정보과학회 2018 한국컴퓨터종합학술대회 논문집, 1869p-1871p 2018.6
[7]http://wiki.dragino.com/index.php?title=Lora/GPS_Shield
[8]http://www.dysystem.net/kor/bbs/board.php?bo_table=200300
[9]https://www.cisco.com/web/KR/events/CiscoConnect/2015/downloads/en/Day1_en2_km_01.pdf
[10]https://fenix.tecnico.ulisboa.pt/downloadFile/3779579626799/Zigbee-sintese.pdf
[11] https://namu.wiki/w/블루투스
[12]이리나, 이가람, 김호원, ”LoRa 기술 분석”, 2017 한국통신학회 하계종합학술발표회 논문집, 217p-218p, 2017.6

본 논문은 한국연구재단 이공학개인기초연구지원사업 (NRF-R1D1A1B07048416)의 연구 결과로 수행되었음

시뮬레이션-이력 데이터 분석 서비스 프레임워크의 설계 및 구현

정현태, 김성현, 서영균
경북대학교 컴퓨터학부
{hathor888, kshy9598, yksuh} @knu.ac.kr

Design and Implementation of Simulation-Provenance Data Analytics Service Framework

Hyun Tae Jeong, Seounghyun Kim and Young-Kyoon Suh
School of Computer Science and Engineering, Kyungpook National University

요 약

최근에 CPU, I/O, 네트워크 등 컴퓨팅 자원의 성능 고도화에 힘입어, 다양한 계산과학공학 분야에서 시뮬레이션을 활용한 연구가 활발히 진행되고 있다. 이에 따라, 다양한 종류의 시뮬레이션 이력에 대한 빅데이터가 축적되고 있다. 그럼에도 불구하고, 계산과학공학 시뮬레이션을 수행하는 사용자들의 편의성을 증대하기 위해 이러한 빅데이터를 활용한 분석 서비스 등에 대한 연구는 아직 미비한 실정이다. 이 시점에서, 본 논문은 계산과학공학 시뮬레이션 사용자들의 시뮬레이션 편의성 증대를 위한 이력 데이터 분석 서비스 프레임워크인, *SPA* (*S*imulation *P*rovenance data *A*nalytics service framework) 시스템을 제안한다. SPA 시스템은 REST API 를 활용한 유연한 결합성 및 다양한 시뮬레이션 보조 기능의 제공함으로써, 계산과학공학 시뮬레이션 사용자에게 편의성과 다양성을 제공할 수 있는 종합적인 프레임워크를 제공한다.

1. 서 론

최근에, CPU, I/O, 네트워크 등 컴퓨팅 자원의 성능 고도화가 급진적으로 진행되었다. 이러한 컴퓨팅 자원의 성능 향상은 기존에 해 볼 수 없었던 다양한 계산과학공학 시뮬레이션의 수행을 가능하게 하였다. 이에 따라, 그러한 시뮬레이션의 수행으로 인한 이력 (provenance) 데이터가 폭발적으로 생성되기 시작했다. 계산과학공학 시뮬레이션의 효율성 측면에서, 생성되고 수집된 시뮬레이션 이력 데이터의 효과적인 활용 및 분석 또한 매우 중요한 시점에 이르렀다.

요즘 한창 각광받고 있는, R[1]과 같은 통계 계산을 위한 프로그래밍 언어 또는 많은 종류의 데이터 분석 및 머신 러닝 라이브러리를 보유한 Python[2]과 같은 도구들을 사용하게 되면, 위에서 언급된 방대한 양의 시뮬레이션 이력 데이터 (즉, 빅데이터)를 자동으로 분석해 볼 수 있다. 그 결과, 우리는 다양한 종류의 시뮬레이션 이력 데이터 분석 서비스를 가능하게 할 수 있음을 알게 되었다. 그럼에도 불구하고, 그러한 계산과학공학 연구자들의 시뮬레이션 편의성을 증대시킬 수 있는 분석 서비스 프레임워크에 대한 연구가 아직 미비한 실정이다.

이러한 시점에서, 본 논문은 온라인 계산과학공학 시뮬레이션 플랫폼인 EDISON[3]으로부터 수집된 이력 데이터를 기반으로 계산과학공학 연구자들의 시뮬레이션 수행의 편의성을 제공하기 위한 이력 데이터 분석 프레임워크인 *SPA* (*S*imulation *P*rovenance data *A*nalytics service framework) 시스템을 제안한다.

SPA 시스템의 주요 특징은 다음과 같다. 먼저, REST API[4]와 JSON 을 활용한, 통합 인터페이스 환경을 지원한다. 이를 통해 EDISON 이외에도 다양한 계산과학공학 플랫폼과 연계될 수 있도록 유연한 결합성을 제공한다. 둘째, 다양한 시뮬레이션 프로그램 (즉, 사이언스 앱(ScienceApp))에서 수행된 기존 시뮬레이션 이력을 활용한 다양한 편의 서비스를 지원한다. 대표적인 서비스로, (i) 주어진 시뮬레이션 파라미터에 대한 시뮬레이션 종료 시간 추정, (ii) 특정 시뮬레이션에 자주 사용되는 파라미터 값들의 집합에 대한 랭킹 서비스 등을 구현하였다.

본 논문의 구성은 다음과 같다. 다음 장은 본 연구와 연관된 관련 연구를 복기한다. 이어, 3 장은 SPA 시스템에 대해 서술하고, 4 장에서 사례 연구 결과를 서술한다. 끝으로, 5 장에서 본 연구의 결론 및 향후 연구에 관해 기술한다.

2. 관련 연구

어떤 연구[5]는 스마트폰으로부터 데이터를 추출하여 서버로 전송한 후 서버에서 데이터를 수집 및 분석하는 기능을 한다. 클라이언트-서버의 형태를 가진

프레임워크라는 점에서 본 논문의 SPA 시스템과 비슷하나, 해당 연구는 프레임워크 내에서 규정한 시뮬레이션에 한하여 시뮬레이션을 진행할 수 있다는 점, 인터페이스를 모바일 애플리케이션의 형태로 제공하는 점, 데이터를 분석하여 통계에 그친다는 점, API 를 제공하지 않는다는 점 등에서 본 연구와 다르다.

다른 연구[6]는 EDISON 플랫폼에서 이루어지는 동일한 시뮬레이션 요청으로 인한 낭비되는 리소스 사용 문제를 극복하기 위해 프레임워크를 제안했다는 점에서는 본 연구와 비슷하나, Elasticsearch[7] 를 기반으로 수집된 실제 시뮬레이션 데이터를 검색을 통해 중복 시뮬레이션 요청 문제점을 극복한다는 점에서 본 연구와 다르다.

또 다른 연구[8]는 IoT 를 통해 수집한 데이터를 분석하기 위한 프레임워크를 논의한다. 데이터를 관리하는 프레임워크라는 점에서 본 연구와 비슷하나, 저자들의 연구는 대량의 데이터를 수집하기 위해 클라우드 컴퓨팅을 사용했다는 점에서 다르다.

앞서 언급된 EDISON 플랫폼은 계산과학공학 시뮬레이션을 온라인으로 실행해 줄 수 있도록 하는 시스템이다. 본 연구는 EDISON 플랫폼에서 수집된 시뮬레이션 이력 데이터를 다루고 있으며, EDISON 에서 제공하지 않는 데이터 분석 서비스를 제공한다는 측면에서 다르다.

본 논문에서 제안하는 SPA 시스템은 Wrapper 기능을 구현하여 사용자가 원하는 시뮬레이션 소스 코드 혹은 툴을 wrapping 하여 사용자에게 여러 가지 시뮬레이션을 제공할 수 있다는 점, REST API 를 이용하여 대부분의 플랫폼에서 사용 가능한 느슨한 결합 형태를 제공한다는 측면에서 차별성을 두고있다.

3. SPA 시스템 설계 및 구현

SPA 시스템은 프레임워크 내에서 스크립트 언어를 통한 시뮬레이션 제공 및 REST API 서버 구축, 웹 인터페이스를 구현 하기 위한 Node js[9]와 데이터를 수집 및 관리하기 위한 MongoDB[10]을 기반으로 한, 그림 1 과 같은 구조로 되어 있다. 시스템은 크게 (1) *Data Manager*, (2) *REST API Request Handler*, (3) *Simulation DataBase*, (4) *Simulation Time Estimator*, (5) *Simulation Query Interface* 로 구성되어 있다.

3.1 Data Manager 와 Simulation Data Base

현재 SPA 시스템은 EDISON 플랫폼에서 제공하는 다양한 시뮬레이션 소프트웨어 데이터 중 2016 년 상반기부터 2017 년 하반기까지 서비스된 나노물리(NANO), 전산의학(CMED), 계산화학(CHEM), 전산유열체(CFD) 분야의 데이터를 사용하였다. 실제 데이터 수는 NANO 약 7 만개, CMED 약 1800 개, CHEM 약 2500 개, CFD 약 36000 개, 도합 약 11 만개의 시뮬레이션 데이터 셋을 사용하였다. EDISON 플랫폼에서 제공된 데이터는 다음과 같이 관리된다.

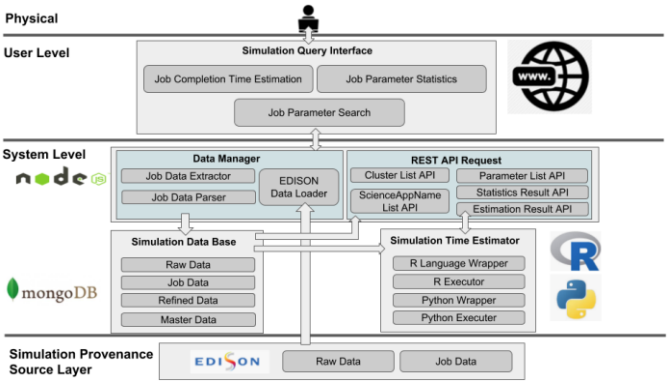


그림 1. SPA 구조

'Data Manager'은 크게 1) *EDISON Data Loader*, 2) *Job Data Extractor*, 3) *Job Data Parser*,로 구성된다. EDISON Data Loader 는 (*Simulation Provenance Source Layer* 에 있는) EDISON 플랫폼으로부터 시뮬레이션 이력 정보 데이터를 원시 데이터 (Raw Data)와 작업 데이터 (Job Data)로 나누어 수집한다. 원시 데이터에는 시뮬레이션 데이터 중 파라미터 셋 이외의 정보이며 작업 데이터는 시뮬레이션 데이터 중 파라미터 셋과 관련된 정보이다.

수집한 데이터는 Simulation Data Base 즉 MongoDB 에 컬렉션 형태로 저장된다. Job Data Extractor 는 작업 데이터로부터 가장 최근 날짜, 즉 최신 버전의 파라미터 값 집합(Parameter Value Set)을 추출하여 Simulation Data Base (즉 MongoDB)에 마스터 데이터(Master Data) 형태로 저장한다. 이렇게 저장된 마스터 데이터는 향후 갱신될 데이터들의 기준 정보로서, 최신 버전으로 유지된다. Job Data Parser 는 EDISON Data Loader 에서 추출된 각 시뮬레이션 이력 정보 데이터를 파싱하고, Simulation Data Base 에 저장된 마스터 데이터를 기반으로, 표 1 의 스키마에 따라 데이터의 형태로 추출하여 정제 데이터 (Refined Data)로 저장시킨다.

3.2 REST API Request Handler

'REST API Request Handler'는 (1) *Cluster List API*, *ScienceAppName List API*, (2) *Parameter List API*, (3) *Statistics Result API*, (4) *Estimation Result API* 등으로 구성된다. 표 2 는 위의 API 에 대한 설명을 제공한다.

3.3 Simulation Time Estimator

'Simulation Time Estimator'는 R 언어와 Python 에 해당하는 Wrapper, Executor 로 구성된다. Wrapper 는 Executor 즉 R 혹은 Python 으로 실행되는 시뮬레이션 툴과 Node Js 기반 서버를 연결하기 위한 인터페이스이다.

표 1. JSON 기반의 시뮬레이션 이력 데이터 저장 및 관리 스키마

`cluster: String`	시뮬레이션 프로그램 (즉 사이언스 앱)이 속해 있는 계산과학공학 분야
`scienceAppName: String`	(특정 계산과학공학분야에 속해 있는) 사이언스 앱 이름
`simulationUuid: String`	시뮬레이션 고유 아이디
`jobExecTime: String`	시뮬레이션 수행 시간
`jobStatus: String`	시뮬레이션 종료 상태
`parameter: Array`	사이언스 앱에 따른 파라미터 리스트
`values: Array`	각 파라미터에 (사용자가) 할당한 실제 값

Wrapper 의 구현 로직은 다음과 같다. Estimation Result API 가 호출 시 Child Process 를 서버 내에서 생성하여 사용자가 입력한 입력데이터를 List 형태로 변화한 다음 Executor 를 호출한 뒤 Executor 에 변환된 List Data 를 제공하며 Executor 에서 반환한 값을 서버에 다시 넘겨주게 된다. Executor 는 Wrapper 로부터 받은 Data 를 기반으로 추정한 Simulation 의 수행시간을 출력으로 가지며 출력 결과는 Wrapper 로 반환하게 된다. 현재 SPA 시스템내에선 R 언어 Wrapper 만을 지원하는 상태이다, R 언어 Executor 즉 실행 시간 추정 서비스에 사용될 머신 러닝파트는 추후 연구 진척에 따라 합칠 것이며, 실시간으로 시간예측 서비스를 지원할 수 있을 것으로 보인다.

3.4 Simulation Query Interface

'Simulation Query Interface'는 (1) *Job Completion Time Estimation*, (2) *Job Parameter Statistics*, (3) *Job Parameter Search* 로 구성된다. Job Completion Time Estimation 은 사용자에게 특정 계산과학공학분야(`Cluster`), 시뮬레이션을 수행하고자 하는 사이언스 앱(`ScienceAppName`), 앱의 파라미터 값 집합(`parameter`)등의 정보를 입력 받아 추정된 시뮬레이션 종료 시간을 웹 인터페이스로 제공한다. Job Parameter Statistics 는 사용자에게 `Cluster`, `ScienceAppName` 을 입력 받아 해당하는 사이언스 앱 내에서 많이 사용된 파라미터 값 집합을 분석하여 10~100 순위까지의 랭킹 결과를 웹 인터페이스로 제공한다. Job Parameter Search 는 사용자가 사이언스 앱을 입력할 경우 해당하는 사이언스 앱이 요구하는 구체적인 파라미터 입력 정보를 웹 인터페이스로 제공한다.

표 2. SPA 시스템에서 제공하는 REST API 설명

API 종류	설명
Cluster List API	Simulation Data Base 에 저장된 마스터 데이터를 기준으로 EDISON 플랫폼에서 지원하는 계산과학공학 분야 리스트를 JSON 의 형태로 제공하기 위한 API
ScienceAppName List API	Simulation Data Base 에 저장된 마스터 데이터를 기준으로 EDISON 플랫폼에서 지원하는 계산과학공학 분야에서 이용 가능한 사이언스 앱 (이름)들을 JSON 의 형태로 제공하기 위한 API
Parameter List API	Simulation Data Base 에 저장된 마스터 데이터를 기준으로 선택된 사이언스 앱이 가지고 있는 파라미터 리스트를 JSON 의 형태로 제공하기 위한 API
Statistics Result API	사용자가 입력한 정보 (계산과학공학분야, 사이언스 앱)를 바탕으로, 정제 데이터에 축적된 데이터를 분석해, EDISON 플랫폼에서 가장 많이 사용된 (즉 사용자로부터 입력된) 파라미터 값 집합의 Top 10~100 을 JSON 의 형태로 제공하기 위한 API
Estimation Result API	Simulation Query Interface 에서 사용자가 입력한 시뮬레이션 실행 정보 (계산과학공학분야, 사이언스 앱, 파라미터 값)를 기반으로, Simulation Estimator 의 R, Python Executor 에 의해 주어진 시뮬레이션의 예상 시뮬레이션 수행시간을 추정하여 밀리초 단위로 제공하기 위한 API

4. 사례 연구

현재 SPA 시스템은 시뮬레이션 이력 데이터를 분석하여 시뮬레이션 시간 추정 서비스와 파라미터 랭킹 서비스를 제공한다. 본 절은 SPA 시스템의 실행 시나리오를 살펴본다.

4.1 시뮬레이션 시간 추정 서비스

실제 EDISON 플랫폼에서 제공하는 시뮬레이션

서비스를 이용할 경우 결과를 도출해내는데 짧게는 3 초의 시간을 필요로 하지만, CF 의 2d_incomp_p 의 경우 230 만초가 소요되는 경우도 있다. 예측할 수 없는 수행 시간은 사용자에게 불편함을 가져다 준다. 시뮬레이션 시간 추정 서비스는 머신러닝을 사용하여 파라미터에 따라 예상되는 실행시간을 미리 추정함으로써 사용자에게 편의성을 제공한다. 시뮬레이션 시간 추정 서비스는 SPA 시스템내의 Simulation Query Interface 중 Job Completion Time 인터페이스를 통해 행해진다. 시뮬레이션 시간 추정 서비스는 Request Handler 에서 Cluster API List, ScienceAppName List, Parameter List, Estimation Result API 가 사용된다. 그림 2 와 같이 현재 구현이 되어 있다.

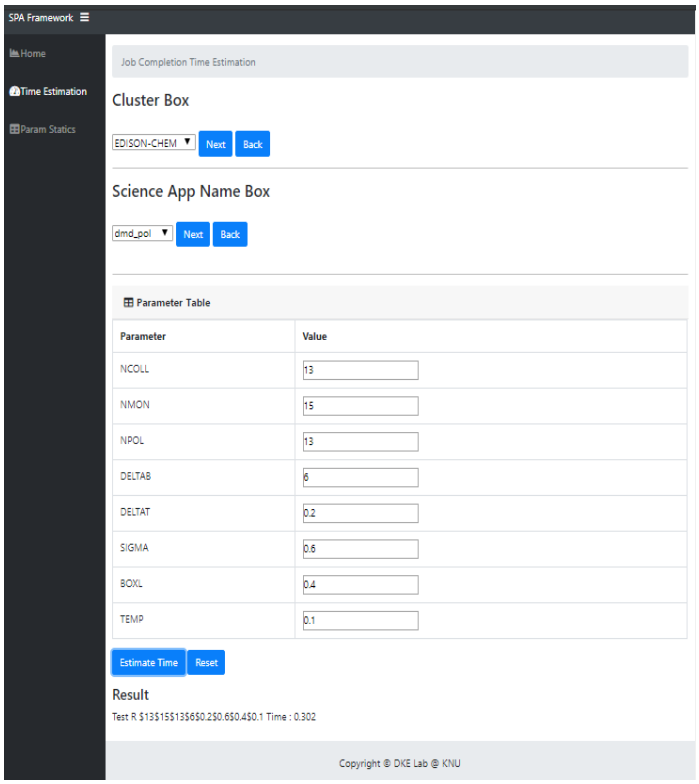


그림 2. SPA 시스템에서 제공하는 시뮬레이션 시간 추정 서비스의 뷰 (View)

그림 2 에서 Cluster Box 는 Cluster List API 로부터 제공받은 EDISON 플랫폼의 사이언스 앱 종류가 리스트에 출력된다. 그후 Science App Name Box 의 Next 버튼을 클릭 시, 특정 계산과학공학 분야에서 제공하는 사이언스 앱 종류가 ScienceAppName List API 를 통해 출력된다. 원하는 사이언스 앱을 입력하고, Science App Name Box 의 Next 버튼을 클릭하게 되면, 선택된 사이언스 앱에 해당하는 파라미터 종류가 Parameter List API 를 통해 테이블 형태로 제공되며, 이후 사용자는 각 파라미터에 대해 대응하는 값을 입력 박스를 통해 입력할 수 있게 된다. 이후, Estimation Time 버튼을 클릭하게 되면, Estimation Result API 가 호출되어 입력 데이터 (예, 분야, 사이언스 앱 이름, 각 파라미터 별 값)를 서버에 전송 후 R Wrapper, R Executor 을 통해 수행 시간을 추정한 후, 결과 값을 하단의 Result 라벨 밑에 출력하게 된다. Reset 버튼을 누르면 모든 상태 값들이 초기화되며, 각 박스에 해당하는 Back 버튼을 클릭 시 이전 상태로 돌아가게 된다.

4.2 파라미터 랭킹 서비스

파라미터 랭킹 서비스는 SPA 시스템에 저장된 EDISON 데이터 중 가장 많이 사용된 파라미터 셋을 MongoDB 의 집계 연산 기능을 이용하여 제공하는 기능이다. 파라미터 랭킹 서비스는 SPA 시스템내의 Simulation Query Interface 중 Job Parameter Statistics 인터페이스를 통해 제공된다. 파라미터 랭킹 시스템 서비스는 Request Handler 의 Cluster API List, ScienceAppName List, Statistics Result API 등이 사용 된다. 그림 3 과 같이 현재 구현이 되어 있다.

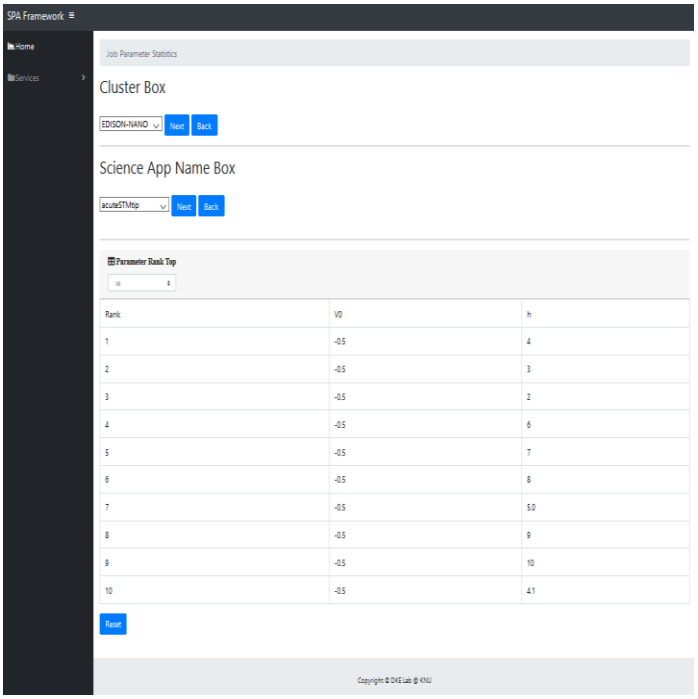


그림 3. SPA 시스템에서 제공하는 파라미터 랭킹 서비스의 뷰

4.1 절의 시뮬레이션 추정 서비스에서 언급된 바와 같이, 그림 3 의 Cluster Box 와 Science App Name Box 는 동일하게 동작한다. Science App Name Box 의 Next 를 클릭 시, 입력 데이터를 기반으로 Statistics Result API 호출하여 사용자가 선택한 사이언스 앱에 대해 가장 많이 사용된 파라미터 셋의 10~100 순위 정보까지 테이블의 형태로 제공한다. Reset 및 Back 클릭 이후, 일어하는 동장 과정은 시뮬레이션 시간 추정 서비스와 동일하다.

5. 결론 및 향후 연구

본 논문은 Node Js 및 Mongodb 를 활용한 시뮬레이션 이력 데이터 분석 서비스 프레임워크인 SPA 를 제안하였다. SPA 시스템은 계산과학공학 연구자들의 시뮬레이션 수행 편의성을 제공하면서 물리적으로 유연하고 가볍게 구현되었다. 기존 연구와는 달리, SPA 내에서 다양한 계산과학공학 분야에서 제공 중인 사이언스 앱에 대한 이력 데이터를 제공할 수 있다는 점, REST API 를 활용해 다양한 인터페이스에서 사용 가능한 것이 가장 큰 특징이다. SPA 시스템에서 제공하는 시뮬레이션 시간 추정 서비스를 통하여 EDISON 플랫폼에서의 수행 시간에 비하여 더 빠르게 시뮬레이션 시간을 도출함으로서 사용자에게 편의성을 제공해주며 파라미터 랭킹 서비스를 통해 사용자가 EDISON 플랫폼에 저장된 데이터를 분석하는데 도움을 줄 수 있다. 이러한 장점을 바탕으로 SPA 는 계산과학공학 시뮬레이션 보조 프레임워크로서 연구자들의 시뮬레이션 효율성 및 편의성을 증대시킬 수 있을 것으로 기대된다.

향후 연구에서, R Executor 기능을 SPA 시스템내에 합침으로써 시뮬레이션 시간 추정 서비스를 완벽히 지원할 계획이며 Python Wrapper 및 Executor 또한 추후 연구 진척도에 따라 완성시킬 계획이다. 또한 우리는 기존 Python, R 뿐만 아니라 Matlab[11]을 지원하여 더 다양한 언어로 이력 데이터 분석을 지원할 계획이다. 또한, SPA 시스템을 사용한 실제 사용자들의 입력 정보 및 기존 시뮬레이션 이력 데이터를 결합하여 더 다양한 차원의 분석 서비스를 지원할 수 있게 할 것이다.

사 사

이 논문은 2017 년 경북대학교 신임교수정착연구비에 의해 수행되었음. 또한 이 논문은 2018 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 신진연구자사업(No. NRF-2018R1C1B6006409), 2018 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No.NRF-2018R1A6A1A03025109), 2018 년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2016-0-00145, 어떤 주제에 대한 빅데이터를 스마트 보고서로 요약하는 기술 개발).

참 고 문 헌

[1] R, https://www.r-project.org/ (2018 년 10 월 확인)

[2] Python, https://www.python.org/ (2018 년 9 월 확인)

[3] Young-Kyoon Suh, Hoon Ryu, Hangi Kim, and Kum Won Cho, “EDISON: A Web-Based HPC Simulation Execution Framework for Large-Scale Scientific Computing Software,” *International Symposium on Cluster, Cloud and Grid* (CCGrid), 2016.

[4] Roy T. Fielding, Richard N. Taylor, Justin R. Erenkrantz, Michael M. Gorlick, Jim Whitehead, Rohit Khard, “Reflections on the REST Architectural Style and ‘Principle Design of the Modern Web Architecture”, *ESEC/FSE’17*, 2017.

[5] 강동주, 조성현, 강신진 “A Framework for Usage Pattern Analysis of Smartphone Applications”, *정보과학회 논문지: 컴퓨팅의 실제 및 레터*, 19 권, 10 호, 7 쪽, 2013-10.

[6] 마진, et al. "계산과학공학 플랫폼을 위한 실행-이력 기반의 시뮬레이션 데이터 관리 프레임워크 설계 및 구현." *인터넷정보학회논문지* 19 권, 1 호, 77-86 쪽, 2018.

[7] Elasticsearch, https://www.elastic.co/kr (2019 년 1 월 확인)

[8] 김병희, “A Study on the IOT(Internet of things) Framework based on Cloud Computing”, *한국통신학회 2014 년도 추계종합학술발표회*. 2 쪽, 2014-10.

[9] Node js, https://nodejs.org/ko/ (2018 년 7 월 확인)

[10] MongoDB, https://www.mongodb.com/ (2018 년 7 월 확인)

[11] Matlab, https://www.mathworks.com (2018 년 8 월 확인)

행정안전부 소프트웨어 보안 취약점 진단기준과 Java 웹 어플리케이션 대상 오픈소스 보안 결함 검출기 검출대상의 총체적 비교

이재훈, 최한솔, 홍신
{21400575, hansolchoe, hongshin}@handong.edu
한동대학교

Systematic and Comprehensive Comparisons of the MOIS Security Vulnerability Inspection Criteria and Open-Source Security Bug Detectors for Java Web Applications

요 약

경쟁적이며 급진적인 오늘날 소프트웨어 개발 산업 현장에 시큐어 코딩 방법론을 효과적으로 적용하기 위해서는 보안 취약점 결함을 자동으로 검출하는 자동 결함 검출기법의 적용이 필수적이다. 본 논문은 Java 웹 어플리케이션을 대상으로 하여 우리 행정안전부가 정의한 42개의 보안 취약점 진단 기준과 총 323개의 오픈소스 보안 취약점 결함 검출기의 검출 대상 결함 패턴을 비교하여, 동일한 결함 패턴을 대상으로 하는 것이 무엇인지를 명시화한 결과를 소개하고, 이에 기반하여 현재 행정안전부 보안 취약점 진단 기준 방법론의 한계점과 발전 과제를 논의한다.

1. 서론

오늘날 정부, 금융 기관 등 주요 사회 기반의 정보 시스템에서 모바일/웹 어플리케이션 형태의 사용자 인터페이스를 제공함에 따라 모바일/웹 어플리케이션 개발 중 발생하는 사소한 소프트웨어 결함이 시스템 전체의 치명적인 보안 문제로 이어지는 사례가 늘고 있다. 이와 같은 모바일/웹 어플리케이션 개발에서의 보안 취약점 발생에 대한 효과적인 대응하기 위해 소프트웨어 개발 및 검토 과정에서 프로그래밍 요소의 안전한 사용 규칙을 규제/강제하는 시큐어 코딩 기반 개발보안 방법론이 실제 소프트웨어 산업 현장에서 적용되고 있는 추세이다. 우리 정부의 행정안전부는 2013년 소프트웨어 보안 취약점 진단 기준을 발표하고 이에 기반 한 보안 전문가를 양성함으로써, 전자정부 소프트웨어의 코드를 보안 전문가가 수검토하여 잠제적인 보안취약점을 진단하는 방법론을 정립해 오고 있다[1].

시큐어 코딩 기반 개발보안 방법론을 개발 주기가 빠른 일반적인 소프트웨어 산업 현장에 효과적으로 적용하기 위해서는 수기적 검토를 대신, 자동화된 정적분석(static analysis) 기반 보안취약점 결함 검출기(bug checker) 사용이 필수적이다. 하지만 보안취약점 검출을 자동적이고 체계적으로 제공하는 상용 도구의 경우, 사용에 많은 비용이 소요되어 일반적인 모바일/웹 어플리케이션 개발상황에서 활용이 비교적 제한적인 실정이다.

상용도구가 가지는 접근성을 해결하는 방법으로 오픈소스 보안취약점 자동검출 도구를 활용한 방법론을 생각해 볼 수 있다. FindBugs, PMD와 같은 오픈소스 보안취약점 자동검출 도구를 우리 정부가 정립한 보안 취약점 진단 가이드라인을 중심으로 한 시큐어 코딩 방법론에 효과적으로 사용하기 위해 다음 두 가지 기술적 난점이 해결되어야 한다:

(1) 현재 우리 정부의 소프트웨어 보안 취약점 진단이 보안 취약점으로 진단하는 대상과 오픈소스 결함 검출기 간의 검출 대상 연관 관계가 명시적으로 파악 되어있지 않음

(2) 검출 대상이 유사한 다수의 오픈소스 결함검출기의 검출 대상 사이의 연관 관계가 명시적으로 파악되어 있지 않음

이러한 문제로 인해, 현재 현업에서는 특정한 종류의 보안 취약점 진단을 위해 어떠한 오픈소스 결함 검출기를 사용해야 하는지 판단하기 어렵다. 또한 복수 개의 오픈소스 결함 검출기를 사용할 경우, 검출대상이 달라 검출 범위를 효과를 증대하는지 혹은 검출대상이 중복되어 실제적인 효과가 없는지를 판별하기 어려운 실정이다.

본 논문은, 오늘날 웹 어플리케이션 개발에 널리 사용되는 Java 프로그램에 대하여 우리 정부가 정의한 42개 보안 취약점 진단 기준이 보안 취약점으로 정의하는 결함과 오픈소스로 개발된 총 323개의 결함검출기가 결함으로 검출하는 대상을 간의 대응 관계를 명시적으로 조사한 연구 결과를 소개한다.

2. 조사 대상과 조사 방법

본 연구는 행정안전부가 발표한 보안 취약점 진단 기준[2]이 제시하는 총 47개 Java 프로그램 보안 취약점 진단 방법과 2018년 6월 현재 4종의 오픈소스 프로젝트를 통해 공개된 총 1310개의 보안 취약점 대상 결함검출기에 대하여, 다음 3가지 연관 관계를 명료화 하는 것을 목표로 하였다:

1. 특정 보안 취약점 진단 방법(이후 진단 방법)이 검출하고자 하는 보안 취약점 결함을 특정한 오픈소스 결함 검출기를 통해 검출할 수 있는가?
2. 특정 보안 취약점 결함검출기가 검출하는 대상은 어떠한 보안 취약점 진단 방법의 진단 대상에 속하는가?
3. 두 개의 보안 취약점 결함 검출기가 검출하고자 하는 대상이 같은가?

본 연구는 2013년 행정안전부(당시 안전행정부)가 발표한 보안 취약점 진단 기준의 7개 카테고리의 총 47개의 보안 취약점 진단 방법 중 (1) Java 프로그램과 연관성이 없는 1개의 진단 방법과 (2) 보안 취약점에 한정되기 보다는 일반적인 결함 유형에 대해 정의한 4개의 진단 방법을 제외한 42개 진단 방법을 조사 대상으로 선정하였다.

본 연구는 Java 프로그램을 대상으로 한 오픈소스 프로젝트 형태의 보안 취약점 결함검출기를 총체적으로 조사하기 위하여, OWASP(오픈소스 웹 애플리케이션 보안 프로젝트)에 소개된 총 41개의 결함검출기 프레임워크의 오픈소스 프로젝트 중 (1) Java언어를 지원하고, (2) 오픈소스 프로젝트로 개발되고 무료로 공개되었으며, (3) 결함검출기의 검출 대상에

본 논문은 소프트웨어중심대학 지원사업(2017-0-00130)과 한국연구재단 신진연구자 지원과제(NRF-2017R1C1B1008159)의 지원을 받은 연구의 결과로 작성됨

대한 설명이 구체적으로 갖추어져있는 FindBugs를 포함하는 FindSecurityBugs[2], PMD[3], LAPSE+[4], SonarQube[5]를 일차적 조사대상 프레임워크로 선정한 후, 선정된 4개의 프레임워크 내에 있는 총 1310개의 결함검출기의 결함검출 기능을 검토하여 보안 취약점과 연관성이 높은 총 323개의 결함검출기를 최종적 조사 대상으로 선정하였다. 본 연구에서는 개발보안 관련 연구 및 소프트웨어 개발에 경험이 있는 1명의 대학원생과 1명의 학부생, 그리고 1명의 교수가 보안 취약점 진단 방법과 결함검출기의 설명, 관련 결함 사례(예: 관련된 CVE, CWE 사례)를 정성적으로 분석하여, 검출 대상 결함이 같은 경우를 찾았다. 우선 모든 보안 취약점 점검 기준과 모든 결함검출기의 설명을 검토한 후, 각 보안 취약점 점검 기준을 중심으로 이와 연관된 결함검출기 집단을 정의하였다. 그 후, 같은 집단에 속한 두 검출기의 명세, 연관 결함 사례, 소스코드를 검토하여, 두 결함 검출기가 같은 대상을 결함으로 검출하는 경우를 찾았다. 이 때, 보안 취약점 진단 방법과 결함검출기는 기본적으로 다대다 연관관계를 가질 수 있으며, 결함검출기 간에도 다대다 연관관계를 가질 수 있음을 전재로 조사를 수행하였다.

3. 조사 결과

본 연구에서는 앞서 선장한 42개의 보안 취약점 진단방법과 323개의 오픈소스 결함검출기 중 동일한 종류의 결함을 검출하는 경우를 파악했다.[2] 표1은 3 상하로 크게 두 부분으로 나누어지는데 1~44행까지는 보안 취약점 진단 방법과 결함검출기 사이의 연관관계 있는 경우에 대한 결과이며, 45~50행은 연관된 행정안전부 보안 취약점 진단 기준이 없는 오픈소스 결함 검출기에 대한 결과이다. 표1의 1~44행 부분에는 7열이 있는데, 첫 번째 열과 두 번째 열은 보안 취약점 점검방법의 커테고리와 구체적인 방법 이름을 보여준다. 세 번째 열에서 여섯 번째 열은 각각 FB+FSB(F), PMD(P), LAPSE+(L), SonarQube(S)에 해당 검토방법과 연관된 결함검출기의 개수 나타낸다. 이 때 괄호 안의 숫자는 해당 보안 취약점 점검 방법에 직접 해당 되지는 않으나, 관련성이 높은 결함검출기의 개수이다. 마지막 열은, 앞선 4개의 열에서 보고하는 결함검출기 중 중복된 것을 제외한, 결함검출 의도가 다른 결함검출기의 숫자이다. 표1의 45~50번째 행은 연관된 보안 취약점 진단방법이 없는 결함검출기에 대한 정보를 보고한다. 이 영역은 총 6개의 행으로 나누어졌으며, 첫 번째 행은 본 연구에서 명명한 결함 패턴 이름이며, 나머지 행은 앞서 설명한 바와 같이, 4개의 결함검출기 프레임워크에서 해당 결함패턴에 대한 검출기의 갯수다.

조사 결과를 바탕으로 다음과 같은 관찰을 할 수 있었다:

1. 총 323개의 조사대상 결함 검출기 중 1개 이상의 보안 취약점에 대응되는 경우는 총 285건이었으며, 총 211종의 고유한 결함 패턴을 검출하였다. 총 323개의 조사대상 결함 검출기 중 38개는 42개의 보안 취약점 진단 기준 중 어디에도 대응되지 않았다.
2. 표1에서 마지막 열이 0인 9개 취약점 진단 방법은 현존 오픈소스 결함 검출기 중 자동적인 진단을 지원하는 방법이 없는 것으로 관찰되었다.
3. 표1의 괄호 안에 표시한 9개의 결함 검출기는, 해당 보안 취약점 진단 기준이 특정하는 진단 대상 결함에 대한 설명에 따를 경우 대응되지 않으나, 진단 기준의 편협한 조건을 개선할 경우 대응 관계를 찾을 수 있는 경우이다.
4. 표3의 46~50행에서는 현재 행정자치부 보안 취약점 진단 기준과는 무관하나 오픈소스 결함 검출기가 개발된 총 38개의 결함 검출기를 5가지 유형으로 분류한 결과로 이들은 대부분이 2017년 이후에 보고된 보안 취약점에 착안하여 개발된 것을 확인할 수 있었다.

[1] 전자정부 SW 개발 보안 진단원을 소프트웨어 개발 보안 약점 진단가이드, 행정안전부, 2013
[2] FindSecurityBugs,https://find-sec-bugs.github.io

[3] PI
[4] LA
[5] Sc

표1. 행안부 보안취약점 진단 기준이 정의하는 보안 취약점을 검출 대상으로 하는 오픈소스 결함검출기의 갯수

(F: FindBugs+FindSecBugs, P:PMD, L:LAPSE+, S: SonarQube)

MOIS Security Vulnerability Weakness Inspection Method		Bug Checkers				
Categ	Security Vulnerability Issue	F	P	L	S	Tot
입력 데이터 검증 및 표현	SQL 삽입	14	0	1	1	16
	경로 조작 및 자원 삽입	10	0	1	1	12
	크로스사이트 스크립트	13	0	1	0	14
	운영체제 명령어 삽입	2	0	1	1	4
	위험한 형식 파일 업로드	2	0	0	0	2
	신뢰되지 않는 URL 자동접속연결	3	0	1	0	4
	XQuery 삽입	9	0	1	0	10
	Xpath 삽입	1	0	1	0	2
	LDAP 삽입	3	0	1	1	5
	크로스사이트 요청 위조	4 (2)	1	0	0	5
	HTTP 응답분할	2	0	2	0	4
	정수형 오버플로우	1 (1)	1 (1)	0	1 (1)	3
	보안기능 결정에 부적절한 입력값 사용	10 (1)	0	2	3	15
	포맷 스트링 삽입	1	0	0	0	1
보안 기능	적절한 인증 없는 중요기능 허용	0	0	0	0	0
	부적절한 인가	0	0	0	0	0
	중요한 자원에 잘못된 권한 설정	0	0	0	0	0
	취약한 암호화 알고리즘 사용	13 (1)	0	0	6	19
	중요정보 평문 저장	0	0	0	0	0
	중요정보 평문 전송	6 (1)	0	0	1	7
	하드코드된 비밀번호	3	0	0	1	4
	충분하지 않은 키 길이 사용	2	0	0	0	2
	적절하지 않은 난수값 사용	2	0	0	1	3
	취약한 비밀번호 사용	0	0	0	0	0
	하드코드된 암호화 키	1	0	0	1 (1)	2
	저장된 쿠키를 통한 정보노출	3	0	0	0	3
	추적문에 포함된 시스템 주요정보	0	0	0	0	0
	솔트 없이 일방향 해시함수 사용	0	0	0	0	0
	무결성 검사 없는 코드 다운로드	1	0	0	1	2
	반복된 인증시도 제한 기능 부재	0	0	0	0	0
시간, 상태	경쟁조건: 검사 시점과 사용 시점	44	6	0	8	58
	종료되지 않는 반복문/재귀함수	3	1	0	3	7
에러 처리	오류 메시지를 통한 정보노출	0	0	0	1	1
	오류 상황 대응 부재	0	6	0	5	11
	부적절한 예외 처리	2	4	0	1	7
캡 슐 화	잘못된 세션에 의한 데이터 정보노출	1	1	0	2	4
	제거되지 않고 남은 디버그 코드	0	0	0	1	1
	시스템 데이터 정보노출	1	0	0	0	1
	Public 메서드의 Private 배열 반환	2	0	0	1	3
	Private 배열에 Public 데이터 할당	1	0	0	0	1
API 오용	DNS lookup에 의존한 보안결정	0	0	0	0	0
	취약한 API 사용	2	6	0	2	10
Uncovered Security Vulnerability Issues		F	P	L	S	Tot
Misuse of Cryptology Algorithms		4	0	0	0	4
Incorrect Use, Framework Features		13	0	0	6	19
Overconsumption, System Resource		1	0	0	2	3
Code Injection		10	0	0	0	10
Log Injection		2	0	0	0	2

[2] https://github.com/GSoJC234/secure-coding-checker-survey

표1. 행정안전부 보안취약점 진단기준에 따른 보안취약점을 검출하는 오픈소스 결함 검출기의 갯수
(F: FindBugs+FindSecBugs, P: PMD, L: LAPSE+, S: SonarQube)

중복된 이슈 보고서 요약

이현권[O1], 감찬규[1], 이선아[1,2]
경상대학교 항공우주 및 소프트웨어공학전공[1], 정보과학과[2]
pppp1476@naver.com, cksrb1526@naver.com, saleese@gmail.com

Summarization of Duplicated Issue Reports

Hyeon-Gwon Lee, Chan-Gyu Gim, Seonah Lee
Department of Aerospace and Software Engineering[1]
Department of Informatics[2]
Gyeonsang National University
ppp1476@naver.com, cksrb1526@naver.com, saleese@gmail.com

요 약

오픈소스 프로젝트에서는 소프트웨어 시스템을 배포하고 이슈 시스템을 유지하는 체계가 일반적이다. 이 때 사용자들은 추가적인 요청 사항을 이슈로 작성하여 보고하며, 다수의 사용자가 보고하는 상황에서는 중복된 이슈 보고서들이 생성된다. 기존의 연구는 하나의 이슈 보고서를 요약하는 기법을 제시하였다. 그러나, 중복된 다수의 이슈 보고서를 요약하는 방법은 제시한 바가 없다. 따라서 우리는 중복된 다수의 이슈 보고서를 요약하는 실험을 진행하고, 이러한 중복 이슈 보고서를 요약하기 위해 어떤 추가 연구를 수행해야 하는지 논의한다.

1. 서 론

소프트웨어 시스템을 배포한 후에 사용자들은 이슈 보고서를 작성하여 추가 요구 사항을 보고한다. 이 때 중복된 내용의 이슈 보고서를 여러 사용자가 작성하는 경우가 발생한다. 이러한 상황은 개발자가 방대한 양의 이슈 보고서를 분석하게 하며 추가 요구 사항 수집을 더 어렵게 한다. 우리는 중복된 다중의 이슈 보고서를 한번에 요약하여 이러한 상황을 돕고자 한다.

기존 연구는 하나의 이슈 보고서를 요약하고자 하는 노력이 있었다[1]. 또한 이러한 초기의 연구[1]가 분류에 있어서 사람의 노력이 많이 든다는 점을 지적하고 사람이 노력이 들지 않는 자동 요약을 제시하는 연구[2]도 있었다. 하지만, 이러한 요약 연구들은 하나의 이슈 요약에 초점을 맞추고 있다.

본 논문에서 우리는 자동 추출 요약 기법을 다수의 중복된 이슈 보고서에 적용하여 요약 결과를 도출한다. 그리고 이렇게 도출된 요약이 다수의 중복된 이슈 보고서의 공통 요약 내용이 될 수 있는 지를 검토하여 보고한다.

본 논문의 구성은 다음과 같다. 2절은 관련 연구를 소개한다. 3절은 실험 모델과 방법에 대해서 설명한다. 4절은 그에 따른 실험 결과를 설명한다. 마지막으로 5절에서 결론을 짓는다.

2. 관련 연구

이슈 보고서 요약과 관련하여 먼저 Rastkar와 Murphy가 제안한 방법이 있다[1]. 그런데 이 제안 방법은 사람이 일일이 태그를 붙여 주어야 하는 단점이 있었다. 후속 연구로 Mani와 동료들은 비지도학습 요약 기법을 비교, 연구하였다. 논문에서는 Centroid, MMR, Grasshopper, DivRank 기법을 비교 실험하였다. 그러나 이러한 연구들은 하나의 이슈 보고서의 요약에 집중한다.

이슈 보고서는 아니지만 다중 문서의 요약에 대한 기법도 있다. 예를 들어 조형락과 동료들은 텍스트 랭크 알고리즘이 문장 사이의 의미적인 관계를 고려하지 않음으로써 무의미한 문장이 대표 요약으로 선정될 수 있는 문제에 주목하였다[3]. 이를 해결하기 위하여 텍스트 랭크 알고리즘 핵심인 그래프 구축 과정에서 정점 간의 관련성을 계산시에 소셜 폭소노미를 활용한 의미적 연관성을 부여함으로써 보다 신뢰성 있는 문서 요약 기법을 제안하였다. 또한 TAC에서 제공하는 정답 요약문과 제안 기법의 요약문을 비교하여 기존 방법에 비해 우수한 성능을 증명하였다.

기타 우리가 사용한 텍스트 랭크 알고리즘을 활용한 연구들이 있다. 예를 들어 트위터처럼 정보 스트림을 이용하는 사용자들은 자신이 구독하는 트위터가 늘어남에 따라 기하급수적으로 늘어나는 정보량을 모두 처리할 수 없는 정보 과다 현상(information overload)을 겪게 된다. 이를 위하여 안인석과 동료들은 텍스트 랭크 알고리즘을 사용하여 트위터의 타임라인을 정리하고 비슷한 키워드들을 하나로 묶어주는 산봉우리 개념이 적용된 알고리즘을 제안하였다[4]. 또한 정의웅과 동료들은 전공 분야의 영어강의 수업 이해도가 낮다는 것을 문제점으로 들며 영어강의 스크립트를 생성하고 핵심을 요약하는 도구를 개발하여 데모를 통해 보여주었다[5]. 그리고 최근 음악이 대량으로 온라인상에 유통되면서 원하는 음악을 찾기 위한 노래

표 1. 요약 대상으로 선정한 이슈 보고서

분류	이슈 번호	이슈 타이틀
User configurable menus	#9285	User configurable menus.
	#9827	Sub menus for Editor Context Menu.
	#9	Commands in context menu
	#11625	Sub menus by extensions in explorer context menus
	#11682	Sub menus by extensions in explorer context menus,item menus
	#12321	Sub menus by extensions in explorer context menus,item menus
User profile	#53453	User profile
	#60755	sync my global setting and installed extension with cloud account
	#56850	Sync VSCode Profile to cloud / login
	#49179	Sync VSCode settings and extensions on the cloud
	#59780	Save and sync extensions to the cloud
New line	#35181	Invisible trailing newline should be the default on OSX and Linux
	#48853	"files.trimFinalNewlines" enabled, doesn't work when file only contains newlines.
Multiple cursor	#55780	Cursor color to distinguish between solitary cursors and groups of cursors
	#6995	Scroll/Reveal when there are multiple cursors
	#11809	Move line up/down should support multiple cursors

가사 요약 기법을 텍스트 랭크 알고리즘과 TF-IDF (Term Frequency - Inverse Document Frequency)로 구현하여 더욱 효과적인 음악 추천 서비스를 제공하고자 한 연구도 있었다[6]. 마지막으로 앱 스토어에 있는 사용자 리뷰를 분류하고 요약하는 연구가 있었다[7].

본 논문은 중복된 다수의 이슈데이터에 텍스트 랭크 알고리즘을 적용하여 요약 결과를 도출하는 실험을 진행하였다는 점에서 기존 연구들과 차이를 보인다.

3. 제안 실험

3.1 실험 모델

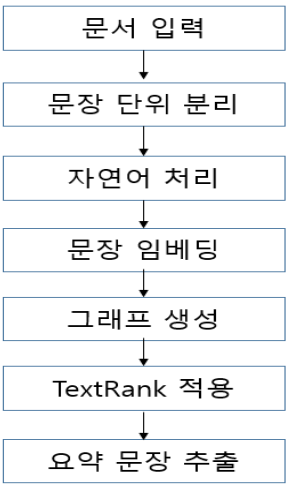


그림 1. 실험 모델 흐름도

본 논문에서 제안한 문서 요약 모델은 크게 입력된 문서 문장 단위 분리, 자연어 처리, 문장 임베딩, 그래프 생성, TextRank 적용, 문장 추출 단계로 이루어져 있으며 모델의 흐름도는 그림 1과 같다.

우선 문장 단위 분리 과정에서는 자연어 처리 라이브러리인 Natural Language Toolkit의 sent_tokenize 함수[8]를 사용하여 문서를 문장 단위로 토큰화 하는 작업을 수행한다. 이후 자연어 처리 과정에서는 문장을 분석하는데 있어 불필요한 불용어를 제거하는 전처리 작업을 수행한다. 여기서 불용어는 nltk에서 정의해서 제공하는 179개의 영어단어를 사용하였다. 다음으로 문장들의 가중치를 계산하기 위해 문장을 벡터화하는 문장 임베딩 단계에서는 문장을 구성하는 단어들을 Glove 데이터 셋[9]을 통해 벡터 값으로 변환하고 산출된 단어 벡터들의 평균 값을 문장 벡터로 나타낸다. 그래프 생성 단계에서는 PageRank를 적용할 문장 기반 그래프 생성을 위한 행렬을 생성하고 행렬을 기반으로 그래프를 생성한다. 생성된 그래프의 정점은 각각의 문장이 되고 정점들 간의 가중치는 문장 간의 유사도가 된다. 제안된 모델에서 유사도는 두 문장의 벡터 간의 코사인 각도를 이용한 코사인 유사도 값이 되며 이를 위해 sklearn 라이브러리의 cosine_similarity 함수[10]를 사용하였다. 마지막으로 생성된 문장 기반 그래프를 TextRank 알고리즘에 적용하여 각 문장의 Rank 값을 계산하고 Rank 값이 높은 순서로 문장들을 추출한다.

3.2.1. 실험 대상

실험 대상 선정 기준은 독자가 보기에 요구사항을 이해하기 쉬우며, 또 현재 많은 사용자가 사용하기

표 2. 이슈 보고서 요약 결과

분류	Rank	추출된 요약 문장
User configurable menus	1순위	One idea is to have a menus.json file which defines your menus.
	2순위	I would like to explorer/context extension right click menu, there is a sub menu , how do I do.
	3순위	The next logical step is to allow user to configure their menus - this is analog to the keybindings-story.
User profile	1순위	I would like to use my outlook account (or other Microsoft account) to save and sync settings and extensions across all my machines running Visual Studio Code.
	2순위	Sometimes I want to run a tool and discover it is not here because I did not yet installed it on this VS Code instance.
	3순위	Also user can create few profiles for example one for working with PHP with needed extensions and theme, another for documentation work with markdown and such.
New line	1순위	All other text editors operating in a Unix-like environment append a newline to the last (or only) line in the file.
	2순위	This is desireable behavior because if you want to use standard Unix tools to work on the file, they will generally expect all lines to end in a newline whether it's printing the file to stdout, combining files, or performing complicated search and replace operations via regular expressions.
	3순위	It follows then that unless the file has no characters, a text file should always end in a newline.
Multiple cursor	1순위	So (through years of using the other editor) I was accustomed to having the multiple cursors go away (for instance, after using an arrow key to move to another line) and through force of habit forgot that VS Code behaves differently in this regard.
	2순위	Since a cursor is almost always in the center of your focal area when you're looking at your editor and typing, it would make sense for every cursor to indicate visually whether it is solo or one of many, rather than for the indicator to be (for example) in the status bar, where it will generally be unnoticed.
	3순위	It would be very useful to know at a glance whether edits you make will only affect the cursor you're currently looking at, or whether there are other cursors (potentially out of the viewable area of the editor) that will have changes made too.
Spell check	1순위	1.Build the node-spellchecker module along with the VSCode and make it available among "standard" modules that extension developers can count upon (this could result in more than one spell checker extension e.g.
	2순위	Even If I would produce node-spellchecker module using node-pre-gyp with binaries for various platforms if I understand things correctly extension cannot (easily?)
	3순위	All three work more or less following a simple idea of loading the dictionary into memory (into a associative table, a.k.a. dictionary, object to be precise).

때문에 분석할 만한 이슈 데이터가 존재하는 프로젝트를 대상으로 하였다. 이에 따라 이슈 데이터를 도출할 대상 프로젝트로 MS에서 개발한 소스 코드 편집기인 VsCode[1] 프로젝트를 선정하였다. 소스 코드 편집기는 이해하기 쉬운 기능을 가지고 있고, 60,841의 이슈 보고서 갯수는 분석하기에 충분한 양으로 보았다.

요약할 이슈 보고서는 중복된 내용을 지닌 다수의 이슈 보고서를 선정해야 하였다. 이를 위하여 두가지 방식을 활용하였다. 먼저 레이블을 검토하여 사용자가 이해하기 쉬운 요구사항을 보고한 이슈를 찾았다. 다음으로는 레이블 중 feature-request 레이블이 있었기 때문에, 이러한 이슈 보고서를 크롤링하였다. 다음 요약할 많은 긴 이슈보고서를 검색하기 위해 바이트 크기로 정렬한 후 큰 사이즈를 가진 이슈 보고서부터 조사하였다. 해당 이슈 보고서의 번호로 검색하여 여러 개의 보고서가 검색 목록으로 보이면, 각각의 보고서를 읽어서 서로 중복되는지 확인하였다. 내용 중복성은 우리가 판단하지 않고 개발자들이 중복성을 답글로 기술한 이슈 보고서를 선정하였다. 그 결과 우리가 요약 대상으로 선정한 이슈 보고서는 표 1과 같다.

3.3 실험 방법

선정된 이슈보고서들의 제목과 내용을 직접 하나의 텍스트 파일에 합치는 작업을 수행한 후 실험 모델에 텍스트 파일을 입력하여 문서 요약 작업을 수행한다. 이후 Rank 값이 높은 상위 3개의 문장을 추출하고 이를 해당 텍스트 파일에 출력하는 방식으로 실험을 진행하였다.

4. 실험 결과

우리는 표 1에서 제시한 이슈들을 요약하는 기법으로 TextRank를 구현, 적용하여 표2와 같은 이슈보고서에 대한 문서 요약 결과를 얻었다.

[1] https://github.com/Microsoft/vscode/issues

이러한 결과 중 우선 추출된 요약 문장이 원래의 이슈 보고서에서의 추가 요구사항을 잘 기술하였다고 자체 평가한 분류는 User configurable menus와 User profile 부분이다. User configurable menus와 관련하여 추출한 요약 문장은 사용자가 메뉴를 구성할 수 있도록 menus.json을 활용하는 방법(1순위), explorer/context의 오른쪽 버튼 클릭 시의 서브 메뉴로의 확장에 대한 요구(2순위), 사용자가 직접 메뉴 탭 구성하는 기능 요구사항(3순위)에 대한 내용을 담고 있다. User profile과 관련해서는 outlook 계정을 통한 VsCode의 사용자 프로파일 설정(1순위)과 확장 기능에 동기화 요구(3순위) 등에 대한 내용을 담고 있다.

도출된 문장이 요구사항을 이해하기에 미진하다고 판단한 분류는 new line, multiple cursor, spell check 부분이다. New line 과 관련된 이슈 보고서에서는 Vscode와 대조되는 Unix 환경에서 작동하는 텍스트 편집기의 new line에 대하여 기술하고 있는데, 이러한 부분을 정확하게 명시한 문장이 없었다. Multiple cursor와 관련된 이슈 보고서들에서는 다중의 커서가 있을 때 스크롤 기능이 안되거나 색상 구분이 안되는 등 다중의 커서를 지원하는 기능에 대한 요청이 있었다. 그러나 1~3순위의 내용을 읽고서 이러한 요구사항을 잡아내기는 쉽지 않았다. Spell check와 관련한 이슈 보고서에서는 스펠링 체크 기능을 탑재하라는 요청이 전반적인 사항이었으나, 추출된 문장들이 이러한 요청보다 상세한 내용을 담고 있어 추가 요구사항을 명확히 확인할 수 없었다.

이러한 실험 결과를 통하여 우리는 단순한 텍스트 랭크의 요약 기법만으로는 다중의 이슈 보고서의 중심 내용을 찾는데 있어서 부족한 결과를 파악하였으며, 이를 기반으로 다중의 이슈 보고서를 요약하는데 적절한 기법을 개발해 나갈 것이다.

5. 결론

우리는 자동 추출 요약 기법인 텍스트 랭크의 기법을 활용하여 여러 개의 중복된 내용을 지닌 이슈 보고서를 요약하는 실험을 진행하였다. 이러한 실험의 결과 중복된 이슈 보고서를 요약하여 공통 요약 내용이 될 수 있는 문장들을 도출할 수 있음을 확인하였다. 그러나 모든 경우에 공통 요약 문장이 추출되지 않았고 또한 추출된 핵심 문장들이 추상적이거나 세부적일 경우 추출된 문장을 통해 해당 이슈를 처음 접하는 개발자가 이슈가 무엇인지 명료하게 알기에 부족하다는 점도 확인하였다. 따라서 우리는 향후 연구로 단어 단위의 도출 및 자동 요약 문장 생성을 연구하여 이러한 부족한 점을 개선할 수 있는 연구를 향후 연구로 진행하고자 한다.

6. 참고 문헌

[1] Rastkar, Sarah, Gail C. Murphy, and Gabriel Murray. "Automatic Summarization of Bug Reports." *IEEE Trans. Software Eng.* 40, no. 4, pp. 366-380, 366-380, 2014.

[2] Mani, Senthil, et al. "Ausum: approach for unsupervised bug report summarization." *Proc. of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, 2012.

[3] 조형락, 김성진, 이동호. “의미기반 텍스트 랭크 알고리즘을 이용한 다중 문서 요약.”, 한국정보과학회 학술발표논문집, , 756-758, 2015.

[4] 안인석, 김현우, 김형주. “텍스트 랭크 알고리즘을 이용한 사용자 타임라인 요약 기법.”, 정보과학회논문지 : 데이터베이스, 39, 4, 238-245, 2012.

[5] 장희웅, 김용준, 이예진, 구명완. “영어 강의 스크립트 추출 및 요약 앱 소프트웨어 개발.”, 한국정보과학회 학술발표논문집, , 1783-1785, 2017.

[6] 손지영, and 신용태. "TextRank 알고리즘을 이용한 음악 가사 요약 기법." 멀티미디어학회논문지 21, no. 1 (2018): 45-50.

[7] Di Sorbo, Andrea, et al. "What would users change in my app? summarizing app reviews for recommending software changes." Proc. of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016.

[8] “NLTK 3.4 Document - nltk.tokenize package” https://www.nltk.org/api/nltk.tokenize.html

[9] Jeffrey Pennington, Richard Socher, Christopher D. Manning. “GloVe: Global Vectors for Word Representation” https://nlp.stanford.edu/projects/glove/

[10] “sklearn.metrics.pairwise.cosine_similarity” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

동의어 치환을 이용한 심층 신경망 모델의 테스트 데이터 생성

이민수 ○ 이찬근
중앙대학교 소프트웨어학부
als950901@naver.com , cglee@cau.ac.kr

Generating Test Data for Deep Neural Network Model using Synonym Replacement

Min-soo Lee○ Chan-Gun Lee
School of Computer Science and Engineering, Chung-Ang University

요 약

최근 이미지 처리 응용을 위한 심층 신경망 모델의 효과적 테스팅을 위해 해당 모델이 올바르게 예측하지 못하는 코너 케이스에 해당하는 데이터를 자동 생성하는 연구가 활발히 진행되고 있다. 본 논문은 문장 분류 심층 신경망 모델에 기반하고 있는 버그 담당자 자동 배정 시스템의 테스트를 위해 입력 데이터인 버그 리포트의 내용에서 임의의 단어를 선택해 동의어로 변형하는 테스트 데이터 생성 기법을 제안한다. 제안된 테스트 데이터 생성 기법을 사용한 경우와 기존의 적대적 테스트 데이터 생성 기법을 사용했을 경우를 다양한 뉴런 기반 커버리지를 중심으로 비교 평가한다.

1. 서론

최근 소프트웨어 테스트 분야에서 화이트박스 테스팅(White-box testing)을 위해 심층 신경망이 예측하지 못하는 코너 케이스(corner case)에 해당하는 데이터를 찾는 연구가 활발히 진행되었다. 기존의 연구는 대부분 이미지 기반 학습 모델의 테스트 데이터인 이미지를 변형시키는 접근을 시도하였다.

본 논문은 이미지 데이터를 기반으로 하고 있는 기존 연구와는 달리 텍스트 데이터를 기반으로 하고 있는 심층 신경망 모델에 대해 동의어 치환을 통해 해당 모델의 코너 케이스에 위치하도록 테스트 데이터를 변형시켜 검증하는 방법을 제안한다. 또한 본 논문에서 검증할 심층 신경망 모델은 [3]에서 개발된 버그 담당자 자동 배정 시스템이다. 해당 시스템은 심층 신경망을 이용하여 기존의 버그 리포트를 해결한 개발자 패턴을 학습하고 새로운 버그 리포트가 주어졌을 때 적절한 개발자를 추천한다.

2. 접근 방법

본 절에서는 텍스트 데이터에 적합한 테스트 데이터 변형 방법에 대해 제안한다. 구체적으로 텍스트 데이터의 변형을 위해 버그 리포트에 나타난 단어들을 사전적 의미가 같은 동의어로 변형시키는 방법으로 진행한다. 워드 임베딩은 Google News-vectors 벡터 테이블을 사용하였으며, 다음 수식 1을 이용하여 버그 리포트에 나타난 단어의 동의어 후보를 결정한다.

$$x^* = x + \lfloor max(L(W,\beta)) \rfloor \quad (1)$$

수식 1에서 x는 변경되기 전 테스트 데이터, x^*는 변경된 후 테스트 데이터이다. $L(W,\beta)$은 x를 입력으로 했을 때, 심층 신경망 모델의 최종 비용이다. 해당 수식은 심층 신경망 모델의 비용을 이용한 테스트 데이터의 변형에 이용된다. 우선 모델을 잘못된 예측으로 유도해야 하기 때문에 $L(W,\beta)$에서 최대인 값을 구한다. 또한 단어인 텍스트 데이터를 변형시켜야 하기 때문에, $L(W,\beta)$에서 최대인 값을 바닥(floor) 함수를 이용해 정수로 만들었다. 생성된 정수만큼 단어들을 선택해 동의어로 변형시킨다. 이렇게 변형된 단어들은 심층 신경망 모델로 하여금 입력 버그 리포트에 대해 적합하지 않은 담당자를 배정하도록 만든다. 전체적인 알고리즘은 그림 1과 같다.

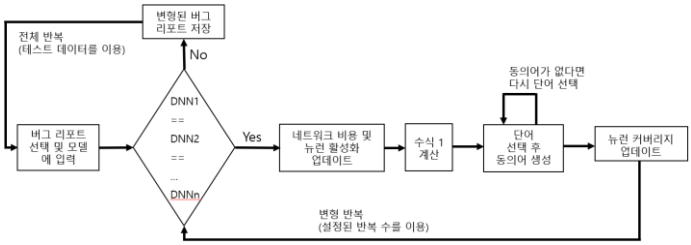


그림 1 제안하는 테스트 데이터 생성 전체 알고리즘

3. 평가 방법 및 실험 결과

3.1 평가방법

본 논문이 제안한 방법이 기존 방법에 비해 효율적인지 평가하기 위해 먼저 TNC, KMNC, SNBC [1, 2]와 같은 뉴런 커버리지 수치를 이용해 기존 이미지를 대상으로 한 차이 유발 테스트 데이터 생성인 GAM, FGSM [1, 2]과 비교하여 보여주는 것으로 평가한다. 그리고 설정된 변형 반복과 전체 반복을 다르게 하여 반복수가 변형에 어떠한 영향을 주는지 SNBC를 이용하여 실험했다.

3.2 실험 결과

본 논문에서 제안한 방법은 “Proposed”라 표기한다.

표 1 차이 유발 테스트 데이터 생성 방법 비교

	TNC	KMNC	SNBC
GAM	0.949	0.947	0.139
FGSM	0.948	0.948	0.165
Proposed	0.951	0.949	0.217

표1은 세개의 생성기법을 적용해서 테스트 데이터를 만들었을 때 다양한 뉴런 커버리지 측정 값을 나타낸다. 이는 변형 반복을 10회로 두고, 전체 반복을 30회로 두었을 때 실험 결과이다.

TNC와 KMNC는 메인 케이스에 위치한 데이터의 커버리지를 측정하는 메트릭이다. 표1를 보면 제안한 방법과 기존의 방법으로 생성한 테스트 데이터가 대략 95% 정도로 학습된 심층 신경망 모델이 예상할 수 있는 영역을 커버했다고 볼 수 있다.

SNBC는 코너 케이스에 위치한 데이터의 커버리지를 측정하는데, 제안한 방법이 GAM, FGSM과 같은 기존 방법보다 대략 3~7% 정도 높은 결과를 보이고 있다. 이는 제안한 방법이 기존 방법보다 3~7% 더 많은 코너 케이스 영역을 커버했다고 볼 수 있다.

표 2 변형 반복과 전체 반복 기반 비교

변형 반복 / 전체 반복	5	10	15	20
10	0.065	0.073	0.098	0.123
30	0.145	0.182	0.196	0.217
50	0.235	0.252	0.259	0.263

표 2는 변형 반복 수와 전체 반복 수를 조정하면서 제안한 방법을 적용할 때 측정된 SNBC의 값을 비교한 실험 결과이다. 첫 번째 열은 변형 반복 수, 첫 번째 행은 전체 반복 수를 의미한다.

전체적으로 변형 반복이나 전체 반복 수를 늘렸을 때 SNBC는 증가하고 있으며, 이는 더욱 많은 코너 케이스 영역을 커버했다는 것을 의미한다. 변형 반복 수를 늘렸을 때와 전체 반복 수를 비교해 보았을 때, 비교적 전체 반복 수를 늘려서 다양한 테스트 데이터를 변형을 진행한 것이 변형 반복 수를 늘려 데이터의 많은 변형을 가한 것보다 SNBC 증가폭이 높다는 것을 알 수 있다. 또한 반복 수가 증가할수록 어느정도 커버리지도 증가한다는 것을 알 수 있다.

4. 결론

이미지를 변형하여 심층 신경망이 예측하지 못하는 데이터를 생성하는 기존 연구와는 달리 본 연구는 제안하는 방법으로 텍스트를 변형하여 예측하지 못하는 데이터를 생성한다. 제안한 방법은 사용하는 심층 신경망 모델의 입력인 버그 리포트의 유의미한 단어를 동의어로 바꾸는 것으로 심층 신경망 모델이 예측하지 못하는 코너 케이스에 위치한 데이터를 생성하는 것이다.

기존 연구의 차이 유발 테스트 데이터 생성 방법보다 본 연구에서 제안한 것이 예측하지 못하는 코너 케이스에 위치한 테스트 데이터를 생성하는 것을 뉴런 커버리지의 관점에서 약 3~7% 좋은 성능을 보이는 것을 확인할 수 있었다. 또한 테스트 데이터의 변형 반복 수, 그리고 전체 반복 수를 나누어 실험하여 전체 반복이 변형 반복보다 SNBC 증가폭이 더 높다는 것을 알 수 있었으며, 반복 수와 뉴런 커버리지가 어느 정도 비례 관계에 있다는 것을 알 수 있었다.

감 사 의 말

본 연구는 한국연구재단 기초연구사업 (과제번호: NRF-2017R1E1A1A01075803)과 한국원자력연구원 원자력 ICT 안전성 검증체계 구축 및 운영과제 (과제번호: 524320-18)의 지원을 받았음.

참 고 문 헌

[1] K. Pei, Y. Cao, J. Yang, and S. Jana, “DeepXplore: automated whitebox testing of deep learning systems,” In Proc. of SOSP, ACM, pp. 1-18, Oct, 2017.

[2] L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu, J. Zhao, and Y. Wang, “Deepguage: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems,” In Proc. of ASE, pp. 120-13, 2018.

[3] 박해성, 김수빈, 이찬근, 채병훈, 딥러닝 모델 기반의 버그 담당자 자동 배정 시스템 성능 측정, 한국 소프트웨어 공학 학술대회 2018.

시장도매인 위탁 판매를 위한 블록체인 기반 시스템 설계

홍준기 ○, 김순태
전북대학교 소프트웨어공학과
[e-mail: { rlwns012, stkim}@jbnu.ac.kr]

Blockchain based System Design for Market-Wholesaler Consignment Sales

Joongi Hong○, Suntae Kim
Dept. of Software Engineering, Chonbuk National University

요 약

우리나라의 도매시장은 대부분 경매제를 통한 거래로 진행된다. 경매 시스템을 통한 거래는 경매 수수료에 의한 유통 비용 증가를 초래하고 당일 농산물 수급 상황에 따라 가격의 등락이 발생한다는 문제가 있다. 대안으로 시장도매인제를 도입할 경우 경매 수수료로 인해 발생한 유통 비용을 낮추고 생산출하자와 협의를 통해 농산물 수급 상황에 따른 가격 변동에 유연하게 대처할 수 있게 된다. 하지만 시장도매인제의 경우 도매시장에서 농산물 위탁 거래가 투명하지 못하여 농산물 생산출하자가 정당한 대가를 받지 못한다는 문제가 있다. 본 논문에서는 시장도매인제에서 문제가 되는 투명하지 못한 거래를 블록체인을 활용한 거래 시스템을 통해 해소하고자 하며, 생산자와 소비자에게 합리적인 거래를 할 수 있도록 지원한다.

1. 서 론

우리나라의 농산물 유통은 대부분 도매시장을 통해 이루어진다. 도매시장은 경매제를 기본으로 하고 있으며, 현재 우리나라 농산물 유통의 절반 가량의 거래량을 차지하고 있는 가락시장 또한 경매제를 활용하고 있다.

경매제는 도매시장에서 입찰 방식을 활용하여 경매를 진행하여 공개된 거래로 공정한 거래가 이루어지며 경매가 끝나는 즉시 농산물 가격에 대한 공개를 하기 때문에 도매시장의 농산물 거래 투명성을 확보했다. 마찬가지로 경매가 끝난 즉시 대금 정산이 이루어지기 때문에 생산출하자들은 믿고 활용할 수 있는 거래 제도이다.

경매제를 활용함으로써 발생하는 단점으로는 경매 수수료로 인한 유통 비용 증가가 있으며, 더불어 경매는 당일 농산물 수급상황에 따른 가격 변동이 크다는 점이 있다. 우리나라의 경우 일반적으로 가락시장의 당일 경매 가격이 농산물의 기준 가격으로 사용되지만 가격이 일정하게 유지되지 못한다는 점을 단점으로 꼽을 수 있다.

서울시농수산식품공사는 경매제의 유통 비용 증가 문제와 당일 수급상황에 따른 농산물의 가격 등락 문제를 해소시키고자 시장도매인제를 추가로 도입하고자 한다. 시장도매인은 수집 기능과 분산 기능이 분리된 경매제와 달리 수집, 분산 기능을 모두 담당하여 거래하는 모델이다. 경매제의 수집과 분산으로 진행되던 유통 과정이 한단계로 합쳐져 유통 비용의 감소를 기대할 수 있고, 동시에 생산출하자와 수의매매를 진행하기 때문에 농산물 수급 상황에 따른 문제에 더욱 유연하게 대처할 수 있다는 장점을 갖고 있다.

시장도매인제를 도입할 경우 경매제의 문제점을 어느정도 해소할 수 있지만 시장도매인은 도매시장의 거래 투명성을 저해한다는 문제를 갖고 있다. 도매시장에서 거래되는 정보를 영세한 생산출하자는 도매시장에서 활동하고 있는 도매인들에 비해 알기 힘들다. 이런 정보의 비대칭으로 인해 시장도매인은 위탁 받은 농산물의 허위 가격 정보를 제공하여 불공정 거래로 이득을 볼 수 있다는 염려가 있어 농산물 생산출하자는 믿고 맡길 수 없는 상황이다.

시장도매인제의 정보의 비대칭으로 인해 발생하는 투명하지 못한 거래를 일정 부분 해소하고 동시에 유통 단계를 축소시켜 유통 비용을 감소시키기 위해서 시장도매인 위탁 판매를 위한 블록체인 기반 시스템 설계를 제안한다. 제안하는 시스템을 활용한다면 생산출하자는 도매시장에 대한 거래 가격 파악이 용이하게 되며 거래를 더욱 투명하게 진행할 수 있다. 또한 시장에서 소비자의 동향을 파악이 수월해짐으로써 농산물의 분배를 시기 적절하게 공급할 수 있게 될 것이라 기대한다. 이 같은 장점으로 생산출하자는 농산물에 대한 정당한 대가를 받을 수 있을 것이며 소비자는 농산물에 대한 더욱 합리적인 가격을 제공받을 수 있을 것이다.

2. 관련 연구

2장에서는 현재 진행이 되고 있는 농업 유통과

블록체인을 활용한 사례와 연구에 대해서 이야기하며, 제안하는 시스템과 어떤 차이점이 존재하는지 알아본다.

2.1 ATFS (AgriTech & Food Science)

ATFS 프로젝트는 “이더리움 블록체인과 스마트 컨트랙트를 기반으로 구축된 세계 최초의 탈중앙화 농업 기술 & 식품 과학 크라우드 펀딩 프로젝트”라고 ATFS 백서에 정의되어 있으며. 스마트 팜의 자금 공급과 농산물 유통을 블록체인과 스마트 컨트랙트를 활용하여 유통 비용 감소와 농업 기술 발전을 이루고자 하고 있다[1]. ATFS의 경우 본 논문에서 가장 핵심적으로 바라보는 도매시장은 고려되지 않은 시스템을 제안하고 있으며, 생산자와 소비자가 직거래를 통해 유통비용을 감소시키고자 하고 있다. 범위 또한 스마트 팜과 가공품 생산까지 이어진다는 점에서 범위의 차이도 존재한다.

2.2 Supply Chain based on Blockchain

Supply Chain[2]은 농산물의 생산부터 소비까지 모든 유통 과정을 Blockchain에 담아 관리하여 생산자와 유통 관계자에게는 농산물의 품질 관리를 수월하게 하며, 유통 비용을 감소시킬 수 있다는 장점을 갖고 있다. 소비자의 측면에서는 유통 과정에 따른 모든 정보를 열람할 수 있게 되어 안심할 수 있는 농산물인지를 확인해보고 믿고 소비를 할 수 있게 된다. Supply Chain의 가장 큰 목적은 농산물의 품질 정보를 위/변조 없이 믿을 수 있는 정보로 제공하는 것이다. 목적 자체가 우리가 제안하는 시스템과는 완전히 다른 시스템이다.

2.3 Blocery

Blocery[3]는 온라인 농산물 직거래 네트워크를 제공한다. 스마트 컨트랙트를 활용하여 거래를 진행하며, 생산자로부터 소비자로 이어지는 최소의 유통 과정을 통해 유통 비용을 감소시키고 생산자와 소비자에게 이득을 줄 수 있는 시스템을 제안한다. 하지만 Blocery도 마찬가지로 도매시장을 대상으로 하는 것이 아닌 생산자와 소비자의 직거래를 지원하는 시스템이기 때문에 본 논문에서 제안하는 시스템과는 차이가 있다.

3. 배경지식

3장에서는 본 논문에서 제안하는 시스템을 이해하기 위해서 현재 우리나라에서 사용하고 있는 농산물 유통 시장의 지식을 필요로 한다. 아래에서는 도매시장과 시장도매인제에 대해 간략하게 소개한다.

3.1 도매시장

도매시장은 농수산물 유통 및 가격안정에 관한 법률(이하 농안법)을 기반으로 정부에 의해 운영, 관리된다. 도매시장은 농안법에 의해 다음과 같은 목적을 취한다.

제1조(목적)
이 법은 농수산물의 유통을 원활하게 하고 적정한 가격을 유지하게 함으로써 생산자와 소비자의 이익을 보호하고 국민생활의 안정에 이바지함을 목적으로 한다. [전문개정 2011. 7. 21.][4]

3.2 시장도매인제

시장도매인이란 도매시장의 개설자로부터 지정 받은 법인을 의미한다. 시장도매인제란 이런 시장도매인이 주체가 되어 농산물을 거래하는 제도를 의미한다. 시장도매인의 농안법에 제정된 법률적 행위는 아래와 같다.

제37조(시장도매인의 영업)
① 시장도매인은 도매시장에서 농수산물을 매수 또는 위탁받아 도매하거나 매매를 중개할 수 있다. 다만, 도매시장 개설자는 거래질서의 유지를 위하여 필요하다고 인정하는 경우 등 농림수산식품부령으로 정하는 경우에는 품목과 기간을 정하여 시장도매인이 농수산물을 위탁받아 도매하는 것을 제한 또는 금지할 수 있다.
② 시장도매인은 해당 도매시장의 도매시장법인·중도매인에게 농수산물을 판매하지 못한다.
[전문개정 2011. 7. 21.][5]

4. 시장도매인 위탁 판매를 위한 블록체인 기반 시스템

[그림 1]은 블록체인 기반 시장도매인 위탁 거래의 개요를 나타낸다. 본 시스템의 가장 일반적인 시나리오는 1단계 농산물 등록, 2단계 위탁 판매 계약, 3단계 소매상인에게 농산물 판매, 4단계 소비자에게 농산물 판매로 총 4단계로 이루어진다.

거래를 위해 필요한 정보와 처리는 모두 Web을 통해 이루어지며 농산물의 인식은 포장에 QR code를 부착하여 인식하고 거래 화면창으로 이동하게끔 구성한다. 농산물의 모든 포장은 농산물 표준규격을 준수하게 끔 되어있으며, 이 상자에 QR code를 부착하여 사용한다고 가정한다.

도매시장은 앞서 말한대로 정부에 의해 운영되고 관리되는 곳이기 때문에 시장도매인제의 시장도매인들이 위탁 판매를 할 때는 모두 본 논문에서 제안하는 시스템을 활용하여 도매시장에서 거래한다고 가정한다. 또한 본 시스템을 활용하기 위해서는 모두 블록체인에 개인키와 공개키를 발급받은 상태이며 이 키를 활용하여 농산물 등록과 거래를 진행하도록 한다.

제안하는 시스템을 이해하기 위해서 다음으로는 각 단계별로 상세하게 어떤 과정을 거쳐 거래 처리가 이루어지는지에 대해서 알아보도록 하겠다.

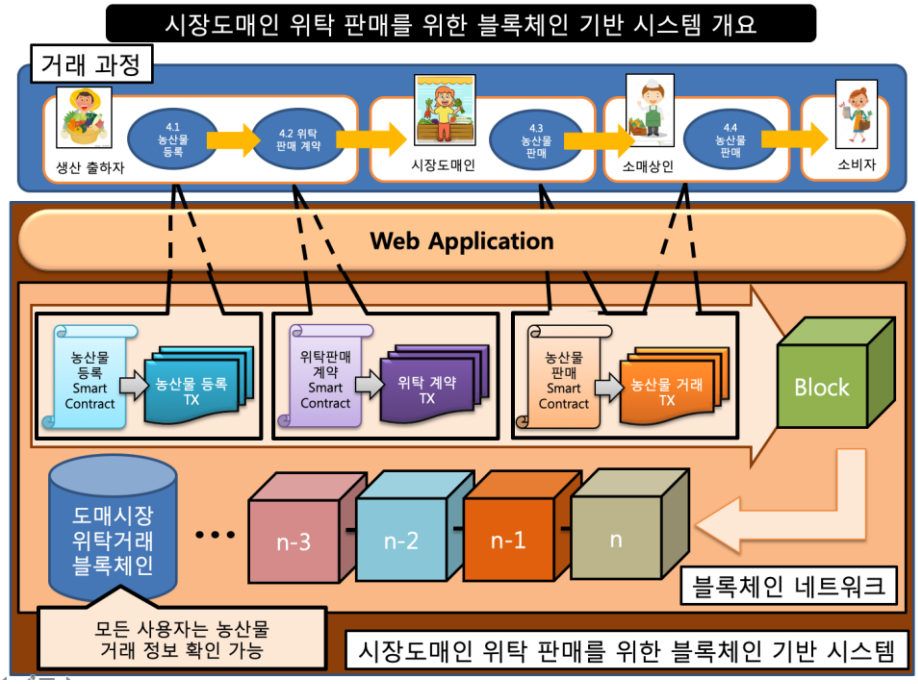


그림 1 시장도매인 위탁 판매를 위한 블록체인 기반 시스템 개요

4.1 농산물 등록

생산출하자는 본인의 농산물을 도매시장에서 거래하기 위해서 농산물 등록 과정을 거친다. 기본 전제로는 농산물은 포장이 되어있는 상태여야 하며, 포장에는 QR code를 부착한다. 부탁 된 QR code는 블록체인 상에서 농산물을 식별하기 위한 식별자로 사용되며 농산물을 추적 관리할 수 있도록 한다.

농산물 등록은 농산물 등록에 관련되 스마트 컨트랙트를 활용하여 진행이 이루어지며 농산물 등록 트랜잭션을 발행하고 블록체인에 담긴다. 농산물 등록에는 농산물의 품목, 생산자 이름, 수량, 가격, 원산지 등 도매시장에서 기존에 거래되던 농산물의 정보를 담게 된다.

4.2 위탁 판매 계약

위탁 판매 계약은 생산출하자와 시장도매인 사이에서 생산출하자가 등록한 농산물을 도매시장에서 위탁 판매하기 위한 계약을 의미한다. 시장도매인은 자신이 도매시장에서 판매하고자 하는 농산물을 제안하는 시스템에서 찾아보고 위탁 판매 스마트 컨트랙트를 활용하여 계약을 진행한다.

계약의 세부 내용으로는 시장도매인은 자신이 위탁 판매를 진행하였을 때 받을 수수료를 입력하고 계약 요청을 한다. 생산출하자는 위탁 판매 요청을 받고 시장도매인의 이름, 판매 내역, 판매 수수료 등의 정보를 제공받고 계약을 진행할 것인지 판단한다. 생산출하자가 자신과 맞는 조건이라고 생각하면 요청에 대해 승낙을 하고 농산물을 시장도매인에게 위탁한다.

4.3 소매상인에게 농산물 판매

세번째 단계에서는 도매시장에서 시장도매인은 위탁 받은 농산물을 소매상인에게 판매한다. 판매를 진행하는 방식은 포장에 붙어 있는 QR Code를 활용하여 정보를 열람하고 소매상인이 실물로도 확인한 후 만족한다면 소매상인은 바로 구매하기 창으로 넘어간다. 농산물에 대한 가격은 이미 생산출하자에 의해 정해져 있기 때문에 소매상인은 구매하고자 하는 농산물의 수량만을 기입하고 스마트 컨트랙트를 이용하여 거래한다. 이렇게 거래된 내용은 모두 블록체인 상에 기록되어 추적이 가능하다.

4.4 소비자에게 농산물 판매

농산물이 소매상인에서 최종 소비자에게 거래가 되는 단계이다. 소매상인도 마찬가지로 농산물을 판매하게 되지만 소매상인은 위탁 판매를 진행하는 것이 아니다. 시장도매인으로부터 농산물을 사온 것이며, 위탁 판매와는 관련이 없다.

소매상은 도매시장에서 나름대로 소비자들에게 농산물 판매를 진행한다. 하지만 거래를 진행하는 방식은 동일하게 QR code를 활용하여 소비자에게 판매하며 소비자도 마찬가지로 농산물에 대한 다양한 정보를 열람할 수 있고, 어떤 생산출하자와 시장도매인을 거쳐 왔는지와 가격이 어떻게 변화하고 현재 해당 농산물이 얼마의 가격을 형성하고 있는지를 확인해 볼 수 있도록 정보를 제공한다.

4.5 제안하는 시스템의 데모 화면

위의 거래 과정에서 존재하는 농산물 등록하기,

농산물 판매의 화면은 아래 [그림 2, 3]와 같이 구성하고자 한다.

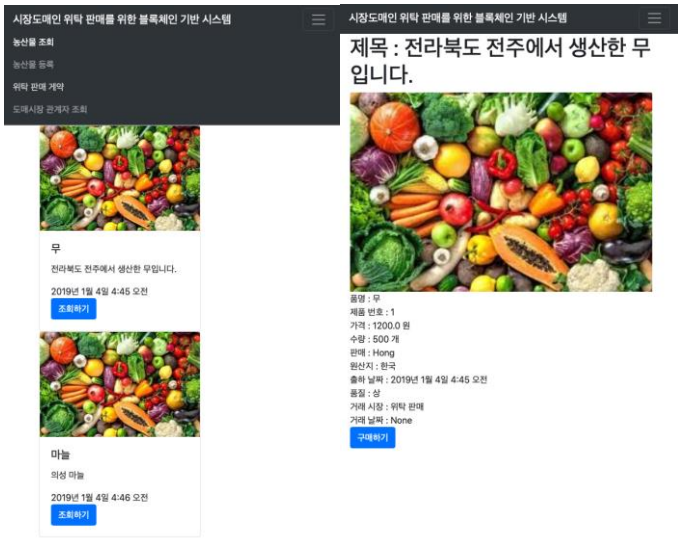


그림 2 농산물 조회 및 상세보기 화면

[그림 2]의 왼쪽은 농산물을 조회하기 위해서 제공되는 창이며 소비자 또는 소매상인들에게 보이는 화면을 구상해 보았다. 오른쪽은 농산물에 대한 상세보기 화면으로 농산물의 품목, 가격, 수량 등 다양한 정보를 열람할 수 있도록 구성하였다.

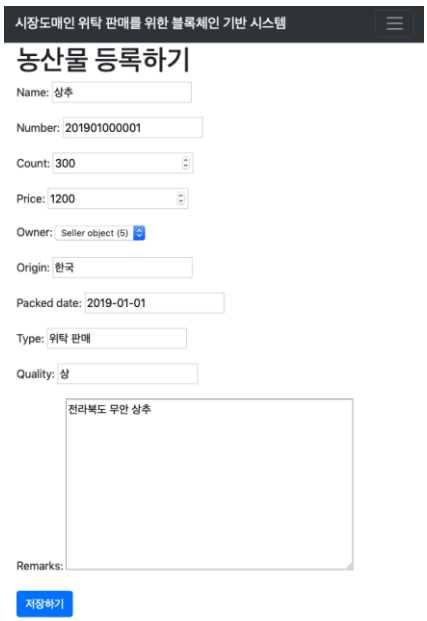


그림 3 농산물 등록 화면

[그림 3]은 생산출하자가 자신의 농산물을 등록하기 위한 화면을 나타낸다. [그림 2]에서 나열된 정보를 이곳에서 기입하게 된다.

5. 결론

본 논문에서는 시장도매인 제도의 정보 비대칭으로 인해 발생할 수 있는 투명하지 못한 거래를 개선하고 유통 단계를 축소시켜 유통 비용을 감소시킬 수 있는 시장도매인 위탁 판매를 위한 블록체인 기반 시스템을 제안한다.

영세 농민은 농산물 유통 시장에서 정확한 정보를 파악하는데 취약하다. 반면에 도매시장 관계자는 농산물의 현재 시세와 소비자의 동향 등을 빠르게 파악할 수 있어 유통 시장 정보에서 우위를 점하고 있다. 이 같은 정보의 비대칭으로 인해 시장도매인은 영세 농민을 상대로 불공정거래를 할 수 있는 문제를 갖고 있어 신뢰하기 힘들다. 본 논문에서 제안하는 시스템을 활용한다면 이런 정보의 비대칭 문제를 어느정도 해소할 수 있을 것이라 판단된다.

정보의 비대칭 문제를 해소하고 투명한 거래를 할 수 있다면 생산출하자는 농산물에 대한 정당한 값을 받을 수 있게 되고, 소비자는 합리적인 농산물 소비를 할 수 있게 될 것이라고 기대하고 있다.

6. Acknowledgement

이 논문은 정부(정보통신기술진흥센터)의 재원으로 대학ICT연구센터육성지원사업의 지원을 받아 수행된 연구임(No. IITP-2018-2017-0-01628).

7. 참고문헌

[1] AFTS Lab, “AFTS White paper”, https://drive.google.com/drive/folders/104R0FqguFGww5eAldm2DZiRpkL-tFjx0, 2018-1 (2018-12 방문)

[2] Tian, F. (2017). A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things. In 14th International Conference on Services Systems and Services Management, ICSSSM 2017 - Proceedings. https://doi.org/10.1109/ICSSSM.2017.7996119

[3] Blocery, “Blocery White paper: Decentralized and Incentive-based Agricultural Product Direct Dealing Service”,https://drive.google.com/file/d/1rn43wgizgcBKm0bGl2lwDy-uzRQgJ2Cb/view, 2018-11 (2018-12 방문)

[4] 농수산물 유통 및 가격안정에 관한 법률, 제1장 총칙, 제1조(목적)

[5] 농수산물 유통 및 가격안정에 관한 법률, 제1장 총칙, 제37조(시장도매인의 영업)

TF-IDF 알고리즘과 API 정보를 활용한 윈도우 실행파일 분류

문재영 ○, 강민구, 김순태, 김강연, 최상철
전북대학교 소프트웨어공학과
[e-mail: {201310500, 201314090, stkim, 201310491, 114477aa}@jbnu.ac.kr]

Windows executable file classification Using TF-IDF algorithm and API information

Jaeyoung Moon, Mingu Kang, Suntae Kim, Gangyeon Kim and Sangchul Choi
Chonbuk National University Department of Software Engineering

요 약

악성코드의 감염은 불법 소프트웨어와 밀접한 관련이 있다. 악성코드에 감염되어 있는 불법 소프트웨어의 사용은 금전적이나 정신적으로 사용자에게 부담을 줄 수 있어, 소프트웨어 필터링을 통한 불법 소프트웨어의 식별이 필요하다. 본 연구에서는 MS 윈도우의 실행파일에서 API 특징정보를 추출하여 프로그램을 분류하기 위한 소프트웨어 분류 기법을 제안한다. 각 API정보들은 특징들을 잘 나타내도록 TF-IDF를 적용하였다. 분류에 적용된 기계학습 알고리즘은 SVM 과 랜덤 포레스트이며, 결과적으로 윈도우 실행파일을 분류하는데 77.2%의 분류율을 보여준다.

1. 서 론

세계 소프트웨어 연합 (BSA | The Software Aliance)은 불법 소프트웨어와 사이버 공격간에 상관관계(r-0.78)가 있음에도 전 세계적으로 불법 소프트웨어의 사용률이 37%를 기록했다고 발표했다. 또한 은행, 보험, 증권 분야에서도 불법 소프트웨어의 사용률은 25%를 기록했다[1]. 불법 소프트웨어의 사용은 예상치 못한 악성코드에 감염될 수 있다. 악성코드들은 피해자의 개인정보나 금융정보를 가져가거나 컴퓨터에 연결된 마이크 및 오디오 장치를 통해 피해자의 사생활을 엿볼 수 있다.

소프트웨어 분류란 프로그램들 사이의 유사성을 찾고, 유사한 프로그램들을 같은 카테고리로 나누는 작업을 말한다. 유사한 프로그램들이란 대표적인

프로그램 기능을 기준으로 비슷한 기능을 가지고 있는 프로그램들이다.

소프트웨어 분류 기법의 사용을 통해 프로그램이 다른 프로그램의 불법 복제 및 사본 여부를 탐지할 수 있다. 소프트웨어 필터링은 소프트웨어가 유통되기전에 불법으로 복제되어 있는 소프트웨어를 식별하여, 사전에 차단하는 방법이다. 의심스러운 프로그램의 기능과 데이터베이스 정보에 저장된 프로그램 기능과 비교하여 일치성을 찾고, 이러한 방법으로 무단 배포를 막을 수 있다. 또한 의심스러운 파일과 원본 파일의 특성을 추출하여 유사도를 파악하여 프로그램을 필터링 처리할 수 있다. 하지만 DB에 존재하는 모든 프로그램과 비교해야 하므로 오버헤드가 큰 문제가 있다. 이런 문제를 해결하기 위하여 소프트웨어 분류 기법을 사용하여 오버헤드가 크게 발생하는 문제를 해결할 수 있다.

기존 연구[5,6,7]는 API 특징과 기계학습을 통한

윈도우 실행파일 분류를 연구하였고 API의 IAT(Import Address Table)을 사용해 분류를 실험하였다. 윈도우 pe .exe파일에서의 API 호출 정보를 추출하고 임계치(Threshold)를 설정하여 데이터를 정제한다.

하지만 기존 연구는 데이터 수집과정에서 파일의 버전을 고려하지 않고, 인접한 버전의 파일을 사용하여 각 API 정보가 카테고리를 대표하지 못한다. 또한, 정제된 API 정보의 공통 API는 시스템 호출 또는 파일 입출력 이므로 API 정보가 카테고리를 대표한다고 보기 어렵다.

본 논문은 각 카테고리별 API 정보와 기계 학습을 이용하여, 윈도우 PE(portable excutable) 실행 파일을 분류한다. 각 소프트웨어 형식은 .exe 파일 형식이며, 각 파일을 IDA Free[8]를 이용하여, API 호출 정보를 획득하고, 이 획득한 정보를 기계 학습에 사용할 수 있도록 임계치를 이용한 API특징 전 처리 방식과, TF-IDF 알고리즘을 고안한 방식을 사용하여 정제한다. 정제된 정보를 가지고 기계학습 알고리즘에 적용하고 분류 결과를 평가한다. 분류 모델로 SVM, 랜덤 포레스트를 사용하였으며, 그 중 랜덤 포레스트 방식이 평균 77% 분류율을 보였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련 연구를 설명하고, 3장에서는 TF-IDF 알고리즘을 고안한 API 정보 활용 윈도우 실행 파일 분류에 대해 설명한다. 4장에서는 실험결과 분석을 하고, 5장에서 최종 결론으로 마무리한다

2. 관련 연구

[6]연구에서 악성코드 실행 시 API 호출 순서를 이용하여 분석하는 방법을 제안하였다. 악성코드가 호출하는 API의 순서를 통해 이를 식별하고, 유사한 악성코드를 탐지할 수 있는 방법을 제시한다. 본 연구에서 악성코드 식별을 위해 API의 사용 여부를 활용한 것과 차이가 있다.

[7]연구에서 제안하는 MDS 기법을 통해 윈도우 실행파일을 대상으로 리소스 영역에서 프로그램의 특징이 될 수 있는 문자열을 이용해 분류를 제안하였다. 추출된 문자들을 버스마크 정보로 사용해 실행파일을 분류하여 오버헤드와 처리시간을 줄일 수 있음을 보여준다. 본 연구는 소프트웨어 분류 기법으로 인한 오버헤드를 줄이는 방법에서 차이가 있다.

기존 연구에서는[5] API정보와 기계학습을 이용해 윈도우 프로그램을 분류하는 작업을 다룬다. 기존 연구에서는 11개의 카테고리로 구성되어 있으며, 각 카테고리당 55개의 PE 파일을 사용해 총 605개의 파일의 API정보를 추출한다. 또한 파일을 정제할 때에는 IDA를 사용해 API 정보 중 IAT를 추출하였고, 그 중 API 이름과 카테고리별 빈도수를 측정한 후 지정된 임계치 보다 많은 프로그램들이 참조하는 API를 특징정보로 선정하여 학습데이터를 이루었다. 정제한 후 기계학습을 통해 분류를 시도하였으며, SVM 알고리즘 방식에서 68%의 분류 효과를 얻었다.

기존 연구[5]에서는 10% 임계치를 기준으로 프로그램에서 사용되는 API 이름으로 분류하였는데 각 카테고리마다 높은 수의 프로그램들이 사용하는 API 이름이 있다. 이들은 공통 API들로 대부분 시스템 호출이나 파일 입출력 등의 동작으로 카테고리를 대표하는 API로 보기 어려움이 있다. 따라서 본 논문은 TF-IDF(Term Frequency - Inverse Document Frequency) 알고리즘 방식을 착안하여, 해당 API를 제거함으로써 특징 차원을 줄여 분류의 정확도를 높이는 방법을 제시한다.

3. TF-IDF 알고리즘과 API 정보를 활용한 윈도우 실행파일 분류

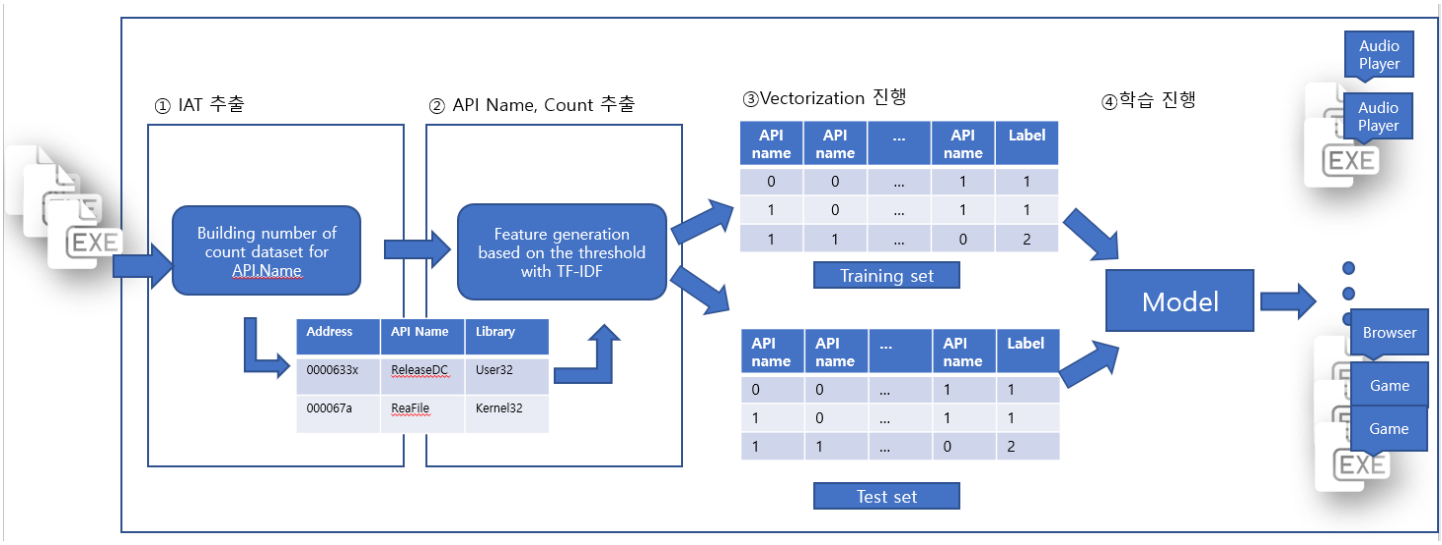
[2]윈도우 실행파일의 주요 정보에는 동적 라이브러리 참조, API export, Import address table, 제어 흐름 데이터, TLS 데이터, 호출그래프 등이 포함 되어있다. 본 논문에서는 API 정보 중 Import Address Table을 윈도우 실행파일을 분류하는데 사용한다

3장에서는 윈도우 실행 파일 분류 방안에 대하여 논의한다.

접근 방법은 그림1과 같이 총 7단계로 이루어져 있다. 첫번째 단계에서는 11개 카테고리 중 각 카테고리별 55개의 PE파일의 IAT를 추출하는 단계이며, 두 번째 단계는 IAT를 추출한 내용을 바탕으로 API name과 카테고리별 Count를 데이터로 가져온다. 세 번째 단계에서는 Feature를 선정하고, data 정의한 후, vectorization 하는 과정을 진행한다. 데이터를 정제하는 과정에서 TF-IDF 알고리즘 방식을 고안하여 사용할 것이며, 차원을 많이 줄인다.

네 번째 단계에서는 Training set과 test set을 구분하여 데이터를 분리한 후 다섯 번째 단계에서 훈련시켜 모델을 만든 후 여섯 번째 훈련된 모델에 test set을 통해 검증을 요구한다. 마지막 일곱 번째 결과를 보여준다. 다음 절에서는 각 단계별 내용을 기술한다

그림 1. API 정보 추출 및 실행 파일 분류 방법 개요



3.1 IAT 추출 및 API name, Count 추출

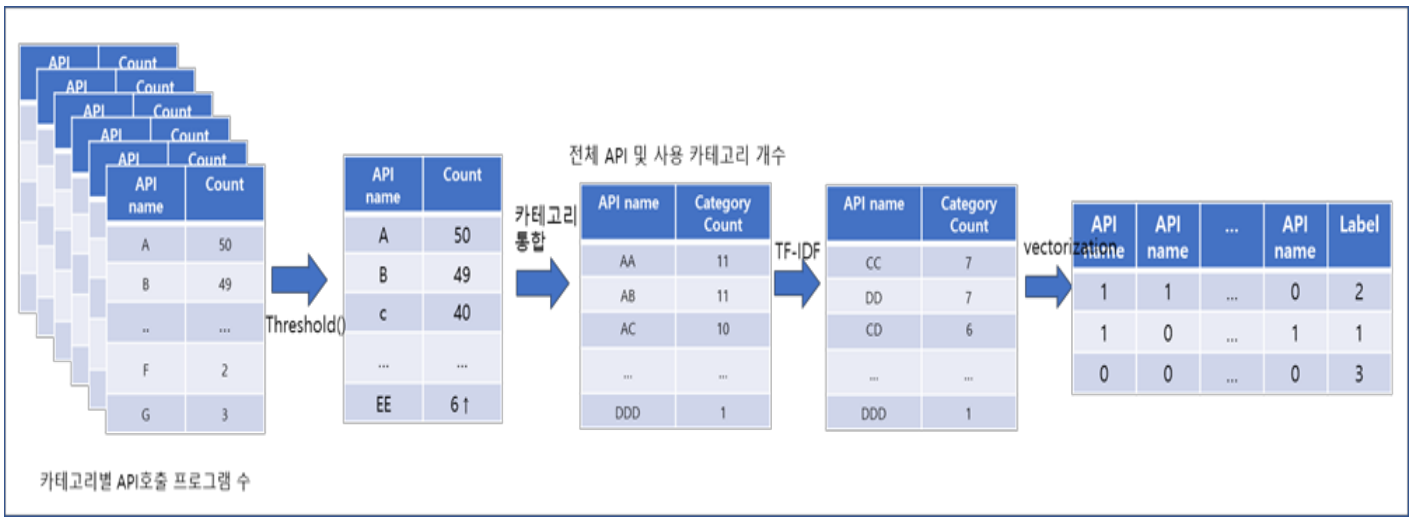
첫 번째 단계에서는 PE 파일을 IDA freeware를 통해 API 정보를 추출한다. 본 연구에서 사용하는 API 정보는 Import Address Table로 호출된 API와 라이브러리 정보를 포함하며, 이러한 정보는 API 사용 여부 판단에 적절하다. 그 중 본 논문은 API name과 각 카테고리별 API name 사용 횟수를 추출한다.

3.2 Feature 선정 및 데이터 정제

그림2는 Feature 선정 및 데이터 정제, 데이터 Vectorization을 표시하였다.

각 카테고리에 있는 55개의 파일의 API 정보를 카테고리 별로 묶어, API 호출 프로그램 수를 수집한다. 그 후 임계치에 의해 Count가 설정한 임계치보다 낮을 경우, 아래 그림 2번처럼 해당 API 정보를 삭제한다. 그 후 11개의 카테고리를 통합하여, API name과 Category Count를 센다. 카테고리 A에서 aa라는 API name이 사용될 수 있고, B에서 aa라는 API name사용될 수 있는데, 이는 같은 API name을 가지고 있기 때문에, 데이터를 정제하는 과정에서 중복을 없애는 방법을 사용해야한다. 본 논문에서는 전체 카테고리 API 정보를 통합할 때, 이를 통해 중복되는 값들을 하기 위해, Category Count를 세어, 중복된 값을 제거한다. 중복된 값이 제대로 정리가 되었을 경우, 데이터를 Vectorization을 통해, 각 파일의 API 정보를 전체 API name에 있어 0과 1로 표시한다. 이러한 단계를 거친 후, 마지막 열에 카테고리에 해당하는 번호를 0~11으로 Labeling을 수행하였다.

그림2. 2번과정 도식화



3.3 기계학습을 이용한 Training set Modelling

실험에서 사용한 기계학습 기법은 SVM과 랜덤 포레스트이다. SVM 학습 기법을 사용할 때에는 비선형 문제 해결할 때, 커널 함수의 영향을 고려하여, 성능에 영향을 파악하고, 선택해야 한다. 또한 Margin을 학습시킬 때, 오류의 허용도에 따라 SVM의 성능이 달라진다. Scikit -learn에서 커널 함수는 선형, 다항식, rbf등이있으며, 본 논문에서는 RBF를 사용하였다. Gamma는 0보다 큰 값을 기준으로 값을 변경하며 실행하였고 Coef 값을 변경하며 테스트를 진행하였다.

두 번째로 사용한 학습 기법은 랜덤 포레스트이다. 랜덤 포레스트는 분류기를 통하여 독립적 학습하는 방법으로, 트리 분류기의 개수에 때라 성능이 영향을 미친다. 따라서 min_samples_split값과 n_estimators의 값을 변경하며 테스트를 진행하였다.

4. 검증

4.1 실험 데이터 수집

데이터는 총 11개의 카테고리에서 55개의 파일로 구성되어 있다.

분류를 하기 앞서, 기존 연구에서는 PE파일로만 이루어진 경우를 이야기했지만, 본 논문에서는 PE파일로 각 다른 파일들을 구성하는 것을 기준으로 하였으나, 만약 기준으로 확보하지 못하는 카테고리일 경우, 1년 이상의 Release 일자 차이가 나는 버전으로 선정하였다.

API 정보는 IDA freeware를 통하여 추출하는데, .NET Framework를 사용한 프로그램이나, 가상 Address로 API가 참조되는 경우 IAT를 확인할 수 없다. 따라서 이러한 파일들은 프로그램을 대체할 수 있는 새로운 프로그램을 확보하였다.

실험에 필요한 테스트 데이터는 기존 연구의 카테고리 11개에서 데이터셋이 불안정한 Audio Player, Messenger 2가지의 카테고리를 제외한 9개의 카테고리에서 각 55개의 파일로 구성되어 있으며, 총 파일은 495개이다. 표에서는 각 카테고리별 프로그램에서 사용된 API의 수를 나타낸다.

표1. 실험 대상 프로그램

Category	Number of Program	Number of APIs (before reduction)
Audio Player	55	21153
Browser	55	6814
CD Write	55	13615
FTP	55	14055
Image Viewer	55	16474
Messenger	55	23345
Text edition	55	21117
Video Player	55	13561
Zip	55	3084
Game	55	3943
Security	55	7414
Total	605	65523(except .dup)

4.2 Feature 선정 및 데이터 정제, 데이터 Vectorization

설정된 Feature를 바탕으로, 표 3과 같이 데이터를 정제하였다. 기존 논문에서는 임계치 7.5, 10%를 사용하였으나, 분석 결과 7.5%에서 API개수가 급격히 증가하여, 10%, 36%의 임계치로 설정하였다.

또한 Category Count를 보면, 많은 수의 카테고리에서 동시에 사용되는 API는 존재하는 것을 볼 수 있다. 이러한 API정보는 그 카테고리를 대표할 수 없다.

본 논문은 카테고리를 대표할 수 없는 데이터는 정제하는 것이 맞으므로, TF-IDF 방식을 고찰해 9개 이상 카테고리에서 사용되는 API 정보를 제거하는 방법을 제시하였다. 임계치가 10%때와 36%였을 때 모두 방법을 시도해보았지만, 36%일 때 API가 30차원으로 너무 축소되는 경향이 보여, 본 논문에서는 10%만을 이용한다.

표3 임계치 기반 정제 후 API 개수

Category	임계치(θ)			TF -IDF + 10% 임계치
	7.5 %	10 %	36%	
Audio Player	1801	1100	120	60
Browser	873	616	180	100

CD Write	1235	1007	412	22
FTP	1653	1309	509	82
Image Viewer	1216	924	389	55
Messenger	2768	1681	363	800
Text edition	2283	976	120	44
Zip	1105	898	459	25
Game	607	388	153	62
Security	1433	1025	366	69
Total	6146	2575	729	1320

4.3 기계학습 별 데이터 및 실험 결과

본 논문에서는 SVM방식과 랜덤 포레스트 방식을 사용하여 기계학습을 시켰다. SVM 알고리즘 방식 중 RBF Kernel을 적용하여서 테스트를 진행하였고. 정규화 및 표준화는 진행하지 않았으며, 임계값 10%와 20%에서 Gamma의 값과 cost의 값을 변경하면서 평가하였다. 또한 랜덤 포레스트 방식의 경우에는 n_estimators의 값과 min_samples_split 값을 변경하면서 평가하였다. 표[4]와 표[5]는 기계학습 기법을 사용한 결과에 대한 평균이다. 테스트 셋과 트레이닝 셋을 5번 이상 변경하여 분류율을 계산한 내용의 평균으로 표를 작성하였다.

표4. 기존 연구 및 본 논문 SVM 방식 프로그램 분류율

Model					
	기존 연구		본 논문		
임계치	7.5%	10%	10%	36	9TF-IDF (10%)
Accuracy	67.9	68.8	46	66.5	72.8
	67.6	68.8	44	64.75	73
	67.6	68.4	46	62.75	71.2
	67.8	68.4	48	62.75	69.2
			42	58.25	68.6
			42	58.25	68.6

표5. 기존 연구[4]및 본 논문 랜덤 포레스트 프로그램 분류율

Model			
	본 논문	기존 연구	
임계치(θ)	10%	7.5%	10%
Accuracy	75.6	67.6	68.4
	76.8	68.8	68.9
	76.4	68.4	67.6
	77.2	68.3	67.8
	77	67.8	67.1
	76.8	67.1	67.6

4. 실험 결과 분석

본 논문의 주제는 기존 연구보다 높은 분류율을 목표로 하고 있다, 따라서 표[4] 표[5]에서 각자 모델들을 기준으로 본 논문과 기존 논문을 분류율을 비교한다.

표[4]는 기존 연구[4]에서 SVM 방식을 이용한 분류율이다. 정규화와 표준화는 진행되지 않았으며, 10% 임계치에서 68.8%의 분류율을 보여주고 있다.

다음 표[4]는 기존 연구에서 사용된 분류율 정보와, 본 논문에서 사용한 10% 임계치 기준으로 한 정제와, 36% 임계치 기준으로 한 정제, 그리고 9가지 이상 카테고리에서 공통으로 사용되는 API 정보 제거한 데이터를 정제하였다. 데이터는 기존의 데이터에서 랜덤으로 테스트 셋과 트레이닝 셋을 분리하여, 진행하였고, 공정한 진행을 위해 4번의 데이터를 새로 만들어 진행하였으며, 평균 값을 나타냈다.

SVM 방식으로 학습한 결과 기존 논문은 68.8%이며, 본 논문은 9TF-IDF 10% 임계치에서 73%로 기존 논문보다 약 4.2% 높은 분류율을 보여주고 있고,

표[5]는 랜덤 포레스트를 이용한 프로그램 분류율이다. 기존 연구에서는 68.9%였고, 본 논문은 77.2%로 전체 표에서 가장 높은 분류율을 보이고 있다.

5. 결론

본 논문은 총 605개의 .PE파일의 API 사용 여부를 추출하여, 기계학습을 이용한 윈도우 exe 실행 파일 분류를 연구하였다. 기존 연구에서는 10겹 교차 검증을 통해 68% 정도의 분류율을 얻었지만, 본 논문에서는 교차 검증을 사용하지 않으면서, TF-IDF 알고리즘을 고려하고, 10% 임계치를 기준으로 정제한 데이터에서 랜덤 포레스트 학습을 이용하였을 때 77.2%로 분류율을 얻을 수 있었다. 이러한 결과로 보아, TF-IDF 알고리즘을 고려해서 많은 문서들이

가지고 있는 API 정보를 제거해 각 카테고리가 가지고 있는 특징들을 잘 나타낼 수 있으며, 이는 높은 분류율로 나타난다.

또한 데이터가 불안정한 11개 카테고리 중 2개의 카테고리를 제외하고 분류를 진행하였음에도 불구하고, 높은 분류율을 얻을 수 있는 것으로 보아, API 사용 빈도보다 API 사용여부가 더 중요하다는 것을 알 수 있다.

6. Acknowledgment

"이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보 ·컴퓨팅 기술개발 사업의 지원을 받아 수행된 연구임(No. NRF-2014M3 C4A7030503)."

7. 참고문헌

[1] BSA Global Software Survey: Sezing Opportunity Through License Compliance]
[2] Microsoft Portable Executable and Common Object File Format Specification. Revision 8.0. Microsoft Corporation
[3] C. Corinna, and V. Vapnik, "Support-vector networks," Machine learning, Vol. 20, No. 3, pp. 273-297, 1995
[4] Breiman, "Random Forests," *Machine Learning*, Vol. 45, No. 1, pp. 5-32, 2001.
[5] 조대희, 임경환, 조성제, 한상철, & 황영섭. (2016). API 정보와 기계학습을 통한 윈도우 실행파일 분류. 정보과학회논문지, 43(12), 1325-1333.
[6] 신강식, 정상문, & 원유재. (2017). API 호출 순서와 파라미터 정보를 이용한 PE 파일 분석에 관한 연구. 한국정보과학회 학술발표논문집, 1086-1088.
[7] 조대희, 박종화, 한상철, & 조성제. (2016). 문자열 정보를 활용한 MS 윈도우 실행파일 분류. 한국정보과학회 학술발표논문집, 117-119.
[8] IDA Support: Freeware Version -Hex-Rays
[9] Robertson, S., “Understanding inverse document frequency：on theoretical arguments for IDF,” Journal of Documentation, Vol. 60, No. 5, 2004, pp. 503-520.
[10] Robertson, S. E., “Term specificity,” Journal of Documentation, Vol. 28, 1972, pp. 164-165.
[11] https://www.softpedia.com/
[12] https://portableapps.com/

의료 영상 데이터 중심 소프트웨어 개발을 위한 학습 노이즈 영상 분류 기법[1]

유동연[O1], 박예슬[1], 이병정[2], 이정원[1]
[1]아주대학교 전자공학과, [2]서울시립대학교 컴퓨터과학과
dongs0125@ajou.ac.kr, yeseuly777@gmail.com, bjlee@uos.ac.kr, jungwony@ajou.ac.kr

Classification of Noise Images Interfering with Learning for Medical Image Data-driven Software Development

Dong-Yeon Yoo[O1], Ye-Seul Park[1], Byungjeong Lee[2], Jung-Won Lee[1]
[1]Department of Electrical and Computer Engineering, Ajou University.
[2]Department of Computer Science and Engineering, The University of Seoul.

요 약

최근 딥러닝 기술의 발전으로, 의료 분야에서도 의료 영상 중심의 보조진단 소프트웨어 개발이 활발해지고 있다. 그중 하나로 딥러닝이나 기계 학습 방식을 적용하여 캡슐내시경 진단을 보조하기 위한 연구들이 있다. 캡슐내시경 영상의 학습을 통해 병변을 자동으로 검출하고자 하고 있으나, 캡슐내시경 영상에는 해부학적인 요소를 관찰할 수 없어 학습에 방해가 되는 노이즈 영상들이 다수 포함되어 있다는 문제가 있다. 이러한 노이즈 영상들은 학습의 성능을 떨어뜨려 좋은 품질의 보조진단 소프트웨어를 보장할 수 없게 된다. 특정 학습 노이즈 영상을 제거하려는 연구들이 진행되었으나, 다양한 학습 노이즈 영상을 한꺼번에 제거하려는 연구는 미비한 상황이다. 또한, 학습 노이즈 영상의 종류마다 노이즈 데이터의 비율이 불균등하여, 학습 노이즈 영상을 그대로 학습하여 제거하는 방법은 제한적이다. 따라서 본 논문에서는 군집화를 통해 다중 노이즈 레이블의 불균등한 데이터를 균등한 데이터로 다시 레이블링하고, 이를 기반으로 학습 노이즈 영상에 대한 통합 분류 기법을 제안하고자 한다. 제안한 기법을 실제 임상 데이터로 실험한 결과, 0.94의 정확도와 0.93의 F1-점수를 보였으며, 데이터의 수가 많은 학습 방해 영상뿐만 아니라 수가 적은 영상도 함께 분류할 수 있음을 확인했다.

1. 서 론

최근 의료 분야에서는 인공지능 기법을 적용한 의료 영상 중심의 진단 소프트웨어 개발이 활발해지고 있다[1]. CT, MRI 등 다양한 종류의 의료 영상 진단 소프트웨어들이 개발되고 있으며, 그중에 하나로 캡슐내시경 영상에 관련된 진단을 보조하기 위한 소프트웨어가 있다[2]. 캡슐내시경 영상은 위장관을 관찰하기 위한 의료 영상으로, 일반내시경 영상과 다르게 입에서 항문까지 위장관 전체를 관찰할 수 있다. 다른 내시경 검사에 비해 촬영되는 시간이 길어, 위장관 내 영상이 보통 5만장 이상이나 촬영된다는 특징도 지니고 있다. 이로 인해, [3]에서는 중복 영상의 문제점을 자각하고, 영상 간의 유사성을 이용한 제거 기법을 제안한 바 있다. 그러나 최근에는 캡슐내시경 영상을 학습하여 진단을 보조하기 위한 시도들이 대부분이다.

학습의 주된 목적은 많은 양의 프레임에서 병변이 존재하는 프레임을 발견하기 위함이다. 예를 들어, 종양을 검출해내기 위해 합성곱 신경망(Convolutional Neural Network)이나 허프 변환(Hough Transform)을 이용한 연구들이 진행되고 있다[4, 5]. 또는 위장관 내 출혈이 발생하면 다른 캡슐내시경 영상보다 붉은색을 띤다는 특징을 이용하여 출혈이 발생한 이미지를 분류하는 연구도 수행된 바 있다[6].

그러나 [4-6]의 연구와 같은 캡슐내시경 영상의 학습 과정에서 모든 프레임이 학습에 사용되면 안된다. 학습을 수행하기 위해서는 촬영된 각각의 프레임에서 나타내고 있는 해부학적 요소에 대해 해석할 수 있어야 하는데, 여러 가지 원인에 의해 모든 프레임이 명확하게 해석될 수 없기 때문이다. 이와 같이 불확실한 영상이 '정상' 혹은 '특정 병변'이라고 레이블이 되어 학습에 사용되면, 각 클래스의 특성을 무디게 만드는 방해 영상이 된다. 예를 들면, 거품이나 이물질이 가득 차서 위장관 점막을 관찰할 수 없는 경우, 혹은 너무 어둡게 촬영된 경우 등이 있다. 본 논문에서는 이와 같은 영상을 '학습 노이즈 영상'이라 정의한다.

'학습 노이즈 영상'을 포함하여 학습하는 경우, 좋은 품질의 소프트웨어를 보장할 수 없게 된다. 많은 연구에서는 이에 주목하여, '학습 노이즈 영상'을 제거하는 방법들을 제안하고 있다. '학습 노이즈 영상' 중 하나인 거품으로 가득 찬 영상을 분류하기 위해서

1) 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00309)

가버 필터(Gabor Filter)를 사용한 연구가 진행된 바 있다[7]. 특히, [7]에서는 캡슐내시경 영상 뷰어를 이용하여 검증을 수행하였는데, 거품 낀 이미지를 제거하여 보여줌으로써 모든 영상을 적재하는 경우에 비해 최대 46% 정도의 시간을 단축했다. 또한 [8]의 연구는 거품 낀 이미지를 분류하기 위한 다양한 알고리즘을 비교하고 평가한 바 있다. 그러나 과거의 연구에서는 '학습 노이즈 영상'이 될 수 있는 다양한 종류의 영상 중에서, 거품 낀 이미지를 제거하는 것에 초점을 맞추어 수행하였다. 따라서 본 연구의 선행연구에서는 다른 '학습 노이즈 영상'의 종류 중 하나인 과잔여물 이미지를 분류하는 연구를 진행한 바 있다[9]. 과잔여물 이미지란 음식물 찌꺼기 등으로 가득 찬 캡슐내시경 영상을 일컫는 말로, [9]에서는 색상 통계 정보를 이용한 SVM(Support Vector Machine) 기반의 분류기를 설계하였다.

하지만 이와 같은 연구들의 한계는 다양한 '학습 노이즈 영상'의 종류 중 하나에 초점을 맞추고 있어, 설계된 분류기들을 이용해서 캡슐내시경 영상에서 발견될 수 있는 모든 '학습 노이즈 영상'을 한꺼번에 제거할 수 없다는 점이다. 따라서 본 논문에서는 모든 '학습 노이즈 영상'을 통합하여 제거할 수 있는 분류기를 설계하고자 한다. 그러나 모든 경우에 대해 학습을 수행할 때, '학습 노이즈 영상'의 종류에 따라 노이즈 데이터의 비율이 불균형하다는 문제가 발생한다. 위장관에서 자주 발견될 수 있는 거품이나 과잔여물의 경우 많은 프레임이 존재하는 반면, 밝기에 의해 관찰이 어려운 경우는 상대적으로 발견되기 어렵기 때문이다. 이를 고려하여, 본 논문에서는 다중 '학습 노이즈 영상'에 대해 더 균등한 형태로 레이블을 생성해줄 수 있는 레이블링 기법 기반의 분류기를 제안한다.

제안하는 기법은 그림 1과 같은 과정을 따라 진행된다. 우선, 비지도 학습 방법의 하나인 K-평균 군집화를 이용하여 다중 '학습 노이즈 영상'에 대해 더 균등하게 분포된 군집을 생성한다. 이렇게 생성된 군집은 분류기를 학습하는 과정에서 활용될 수 있는 레이블이 된다. 다음은 레이블링된 노이즈 영상에 정상 이미지를 추가하여 '학습 노이즈 영상'의 제거를 위한 분류 학습을 수행한다.

제안하는 방법은 실제 환자 11명의 임상 데이터를 이용하여 평가되었다. 평가 방법은 군집의 개수(k)에 따라, 학습된 분류기의 성능을 비교하여 수행되었다. k=4로 군집화한 경우 0.94의 정확도와 0.93의 F1-점수의 가장 좋은 성능을 보였으며, 데이터 수가 적어 학습될 수 없었던 '학습 노이즈 영상'까지 모두 분류될 수 있음을 확인하였다.

결과적으로 본 연구는 의료 보조진단 소프트웨어 개발 시 실제 환자의 임상데이터를 기반으로 학습에 방해가 되는 노이즈 영상을 다양한 각도로 제거하는 기법을 제안함으로써, 좋은 품질의 데이터가 높은 정확도를 가지는 분류기를 만들 수 있음을 보였다.

2. 관련 연구

2장에서는 '학습 노이즈 영상'을 제거하기 위한 기존의 연구들을 소개한다. 먼저, '학습 노이즈 영상' 중 많은 비중을 가지고 있는 거품 낀 이미지에 대한 연구가 있다. [7]의 연구는 캡슐내시경 영상에서 발견할 수 있는 거품들의 크기가 각기 다른 타원형이라는 점을 이용하여 거품 낀 이미지를 분류하는 방법을 제안하였다. 16개의 가버 필터를 이용하여 타원형 거품들을 검출하고, 거품이 이미지의 50% 이상을 차지한다면 거품 낀 이미지로 분류하였다. 캡슐내시경 영상 뷰어를 통해 검증을 수행하였으며, 모든 영상을 적재하는 경우에 비해 최대 46% 정도의 시간이 단축하였다. 가버 필터 이외에도 다양한 방법을 이용한 연구들이 진행되었고,

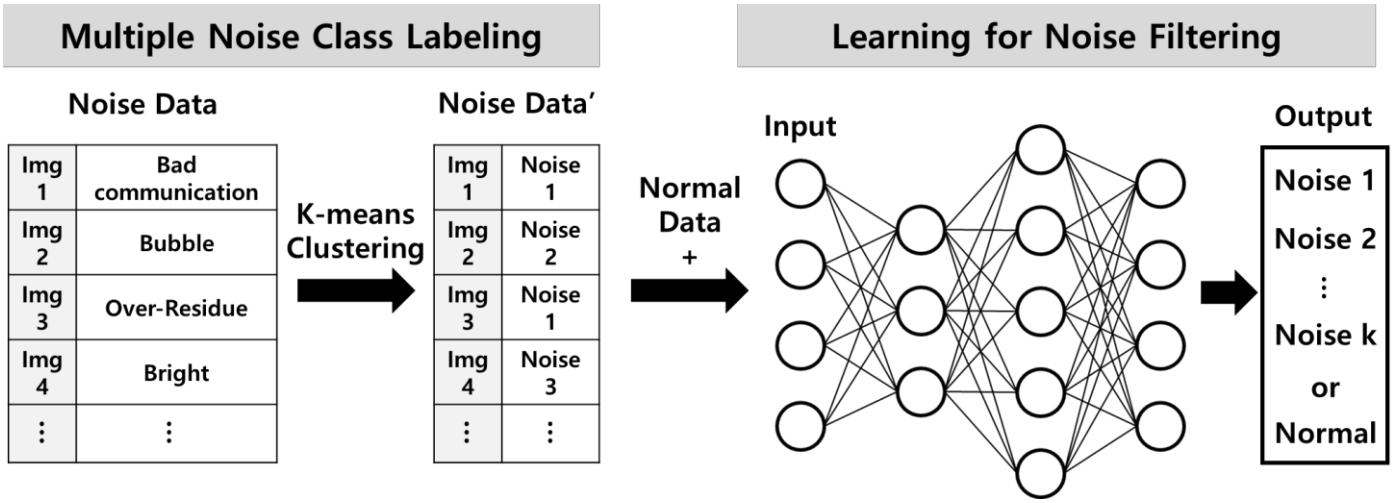


그림 1. 다중 학습 노이즈 영상의 분류 과정

이러한 연구들이 제안한 방법을 분석한 연구도 진행된 바 있다[8]. 해당 연구에선 4가지 분류 방법을 비교하였으며, 그중에서도 GLCM(Gray-Level Co-Occurrence Matrix) 방법은 계산 시간, 정확도, 재현율, 정밀도 모두 다른 방법들에 비해 좋은 성능을 보였다.

다음은 거품 낀 이미지와 대등하게 많은 비중을 차지하고 있는 과잔여물 이미지에 대한 연구다. 학습 가능한 정상 이미지가 대게 붉은빛을 띠고 있는데 반해, 과잔여물 이미지는 음식물 찌꺼기 등의 이물질 때문에 황색 혹은 녹색을 띠는 특징이 있다. 이에 착안하여, 본 연구의 선행연구에서는 색상 통계 정보를 이용하여 과잔여물 이미지를 분류하는 방법을 제안하였다[9]. 캡슐내시경 영상의 특징을 추출하기 위해 영상을 크기가 다른 5장의 이미지로 자른 뒤, RGB 색상 별로 통계 정보를 계산하였다. 색상 통계 정보는 과잔여물 이미지와 정상 이미지를 구분할 수 있는 근거가 되며, 선형 SVM 기법을 적용하여 과잔여물 이미지를 분류하였다. [9]의 연구도 제안한 방법으로 실험을 여러 번 반복한 결과, 평균 0.95의 정확도와 0.89 F1-점수라는 비교적 높은 성능을 보였다.

하지만 기존의 연구들은 특정 '학습 노이즈 영상'을 대상으로 정상 이미지와 구분하는 이진 분류를 시행했다. 앞서 언급한 바와 같이 모든 '학습 노이즈 영상'을 한꺼번에 제거할 수 없으며, 동시에 분류하려 하더라도 '학습 노이즈 영상'마다 노이즈 데이터의 수가 불균형하기 때문에 학습의 성능을 보장할 수 없다는 문제가 있다. 따라서 본 논문에서는 다중 '학습 노이즈 영상'을 균등한 분포로 레이블링한 뒤에 학습하는 '학습 노이즈 영상' 분류 기법을 제안한다.

3. 다중 학습 노이즈 영상 분류 기법

3장에선 다중 '학습 노이즈 영상'을 분류하는 방법에 대해 설명한다. 3.1절에선 다중 '학습 노이즈 영상'을 이용하여 레이블링하는 과정에 대해 설명한다. 그리고 3.2절에선 재분류된 다중 '학습 노이즈 영상'을 학습함으로써 불균형한 레이블도 함께 분류할 수 있는 방법에 대해 설명한다

3.1 K-평균 군집화 기반 다중 학습 노이즈 영상 레이블링

3.1절에선 군집화를 통해 다중 '학습 노이즈 영상'을 레이블링하는 과정에 대해 설명한다. 우선, 학습에 방해가 될 수 있는 영상의 종류는 그림 2와 같으며, 각각의 영상 대한 설명은 다음과 같다.

- Normal Image: 위장관의 해부학적인 구조를 관찰할 수 있는 영상. '학습 노이즈 영상'에 포함되지 않으며, 학습 데이터로 사용할 수 있는 영상

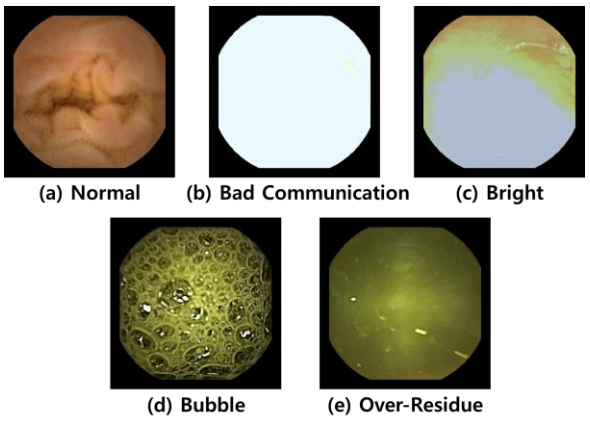


그림 2. 학습 노이즈 영상의 종류

- Bad Communication Image: 통신 장애로 인해 잡음이 섞인 영상
- Bright Image: 조명장치의 문제로 인해 너무 밝게 촬영된 영상
- Dark Image: 조명 장치의 문제로 인해 너무 어둡게 촬영된 영상
- Blur Image: 조명 장치의 문제로 인해 너무 흐리게 촬영된 영상
- Bubble Image: 거품으로 가득 차서 위장관의 해부학적인 구조를 관찰할 수 없는 영상
- Over-Residue Image: 음식물 찌거기로 가득 차서 위장관의 해부학적인 구조를 관찰할 수 없는 영상

2-(a)의 Normal 이미지를 제외한 나머지 이미지는 모두 '학습 노이즈 영상'으로 분류된다. '학습 노이즈 영상'중에서도 2-(d) Bubble 이미지와 2-(e) Over-Residue 이미지는 캡슐내시경 영상에서 많은 비중을 차지하고 있다. 하지만 2-(b) Bad Communicaion 이미지나 2-(c) Bright 이미지는 상대적으로 적게 촬영되어, '학습 노이즈 영상'의 데이터 분포는 균등하지 않다. 본 논문에서는 Bright 이미지와 같이 불균형한 데이터를 갖는 다중 노이즈 레이블을 더욱 균등한 분포의 레이블로 변형해주는 K-평균 군집화 방법 기반의 레이블링 기법을 제안한다. 레이블의 개수는 K-평균 군집화에 사용되는 k에 따라 변화하며, 기존의 '학습 노이즈 영상'은 k개의 노이즈 영상으로 재분류된다. 그림 3은 '학습 노이즈 영상'의 레이블링 과정에 대한 설명이다. 군집화를 하기 위해서 RGB 색상 별 통계 정보를 영상의 특징으로 추출한다[9]. 우선, 캡슐내시경 영상을 크기가 다른 5장의 이미지로 잘라낸다. 잘린 5장의 이미지마다 RGB 색상 별 평균 및 표준편차를 계산하여 하나의 벡터를 생성한다. 마지막으로 영상마다 생성된 벡터들을 이용하여 K-평균 군집화를 수행한다. 군집화로 재분류된 노이즈 영상을 학습 데이터로 사용하여, '학습 노이즈 영상' 분류를 위한 학습을 수행한다.

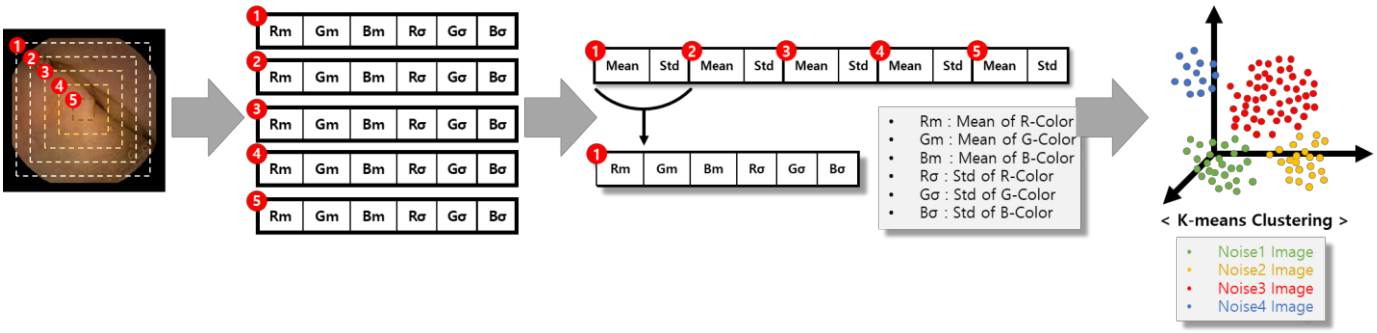


그림 3. K-평균 군집화를 통한 다중 학습 노이즈 영상의 레이블링 과정

3.2 레이블링된 다중 학습 노이즈 영상 분류

3.2절에서는 군집화 과정을 통해 k개로 레이블링된 '학습 노이즈 영상'에 학습 가능한 영상을 더해, '학습 노이즈 영상' 분류를 위한 학습을 수행한다. 구체적으로 설계된 분류기를 학습하는 과정은 그림 4와 같다. 이는 [10]에서 제안한 잔여 네트워크(Residual Network)의 구조를 활용한다. 학습의 결과는 군집화에 사용된 k에 따라, k+1개의 출력층 레이블 중 하나로 나타난다. 학습의 결과가 노이즈 레이블 중 하나로 나타난 경우 '학습 노이즈 영상'으로, 정상 레이블로 나타난 경우 학습 가능한 영상으로 구분할 수 있다.

4. 실험 및 평가

4장에선 3장에서 제안한 '학습 노이즈 영상' 분류 방법을 실험을 통해 평가한다. 실제 환자 11명의 캡슐 내시경 영상을 실험에 사용했으며 표 1은 실험에 사용된 데이터에 대한 설명이다. 약 3.2만장의 캡슐내시경 영상을 실험 데이터로 사용하였으며, '학습 노이즈 영상' 중

표 1. 실험에 사용된 캡슐내시경 영상 데이터

Patient No.	Bubble	Over-Residue	Bright	Bad Com-munication	Normal
1	-	1,716	-	1	1,717
2	13	793	-	2	808
3	38	2,996	11	-	3,045
4	-	772	-	-	772
5	123	640	-	-	763
6	3	5,762	-	-	5,765
7	36	101	-	-	137
8	170	12	-	-	182
9	-	1,426	-	23	1,449
10	122	14	-	-	136
11	-	591	383	-	974
Total	505	14,823	394	26	15,748
	15,748				

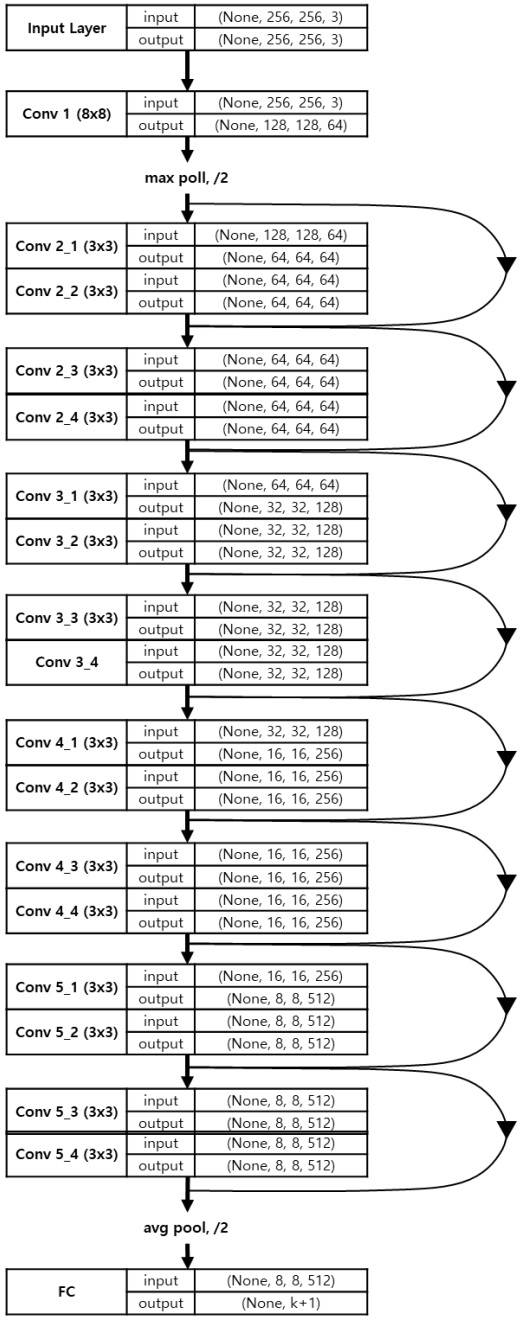


그림 4. 학습에 사용된 잔여 네트워크의 구조

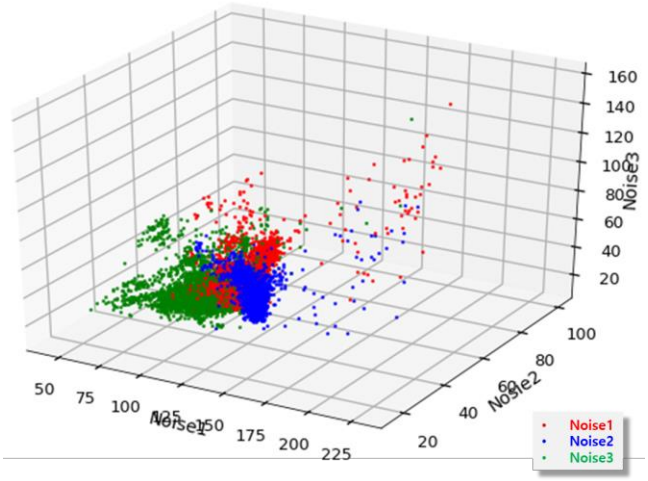


그림 5. K-평균 군집화(k=3)를 통해 재분류된 학습 노이즈 영상

에선 Over-Residue 이미지가 가장 많은 비중을 차지하고 있다. 표 1의 데이터를 이용하여 K-평균 군집화를 수행하였으며, k=3인 경우의 군집은 그림 5와 같이 나타날 수 있다. 그림 5는 영상의 특징을 추출하기 위해 자른 5장의 이미지 중에서 가장 큰 이미지의 RGB 평균값을 이용하여 그린 그림이다. 그림에 표시된 RGB 색상의 점들은 각각 노이즈 1-3 레이블로 재분류된 '학습 노이즈 영상'을 의미한다.

k 값에 따라 군집된 결과는 그림 6과 같다. 그림 6을 살펴보면 k가 2 혹은 3인 경우에는 inertia 값이 급격하게 감소하지만 k가 4 이후부터는 비교적 완만하게 감소하는 것을 확인할 수 있으며, 본 논문에서는 k를 3-6으로 설정하여 레이블링된 데이터들로 학습을

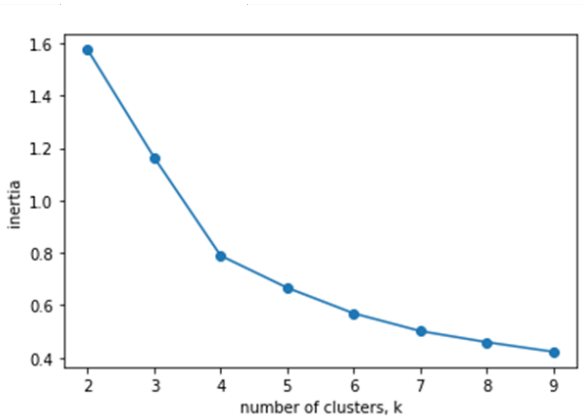


그림 6. K-평균 군집화 과정에서 설정한 k 값에 따른 inertia 값

수행하였다. 캡슐내시경 영상 데이터 중에서 50%를 학습 데이터, 30%는 검증 데이터, 나머지 20%는 평가 데이터로 사용하였으며, 표 2는 k에 따라 측정한 정확도, 재현율, 정밀도, F1-점수를 나타낸다.

k=4로 설정하여 군집화를 수행한 경우, 평균 0.94의 정확도와 0.93의 F1-점수를 보여 다른 k를 사용했을 때보다 성능이 좋음을 확인할 수 있다. k가 4보다 작은 경우에는 기존의 '학습 노이즈 영상'의 레이블의 개수보다 더 적어 낮은 성능을 보이는 것으로 예상된다. 반면에 k가 4보다 큰 경우에는 기존의 '노이즈 영상'보다 많은 수의 레이블로 분류되어, 비교적 낮은 성능을 보이는 것으로 예상된다.

k=4로 설정하여 군집화를 수행한 경우, 평가 데이터에는 Bubble 이미지 101장, Bright 이미지 79장, Bad Communication 이미지 5장이 포함되었다. 기존의

표 2. 레이블링된 다중 학습 노이즈 영상을 이용한 학습 결과

k	valuation	Normal	Noise1	Noise2	Noise3	Noise4	Noise5	Noise6	Average
3	Accuracy	0.8	0.71	0.97	0.98	-	-	-	0.97
	Recall	0.65	0.69	0.95	0.98	-	-	-	0.82
	Precision	0.87	0.88	0.99	0.91	-	-	-	0.92
	F1-score	0.74	0.77	0.97	0.94	-	-	-	0.96
4	Accuracy	0.92	0.96	0.95	0.92	0.93	-	-	0.94
	Recall	0.95	0.91	0.94	0.91	0.93	-	-	0.93
	Precision	0.89	0.9	0.92	0.97	0.99	-	-	0.93
	F1-score	0.92	0.9	0.93	0.93	0.96	-	-	0.93
5	Accuracy	0.86	0.81	0.86	0.83	0.89	0.92	-	0.86
	Recall	0.81	0.89	0.82	0.85	0.85	0.97	-	0.87
	Precision	0.92	0.75	0.88	0.93	0.84	0.88	-	0.87
	F1-score	0.86	0.81	0.85	0.89	0.85	0.93	-	0.86
6	Accuracy	0.79	0.84	0.83	0.86	0.85	0.92	0.95	0.86
	Recall	0.84	0.94	0.81	0.92	0.94	0.95	0.92	0.90
	Precision	0.61	0.7	0.93	0.75	0.81	0.79	0.94	0.79
	F1-score	0.71	0.8	0.87	0.83	0.87	0.96	0.93	0.84

‘학습 노이즈 영상’ 별로 정확도를 계산한 결과, Bubble 이미지 0.86, Bright 이미지 0.92, Bad Communication 이미지 1.00의 정확도를 보였다. 그림 2의 예시에서 확인할 수 있듯이 Bright 이미지와 Bad Communication 이미지는 하얗거나 밝게 촬영되어 다른 레이블의 영상들과 확연한 차이가 있기 때문에 높은 정확도를 보이는 것으로 예상된다.

5. 결 론

본 논문에서는 다중 ‘학습 노이즈 영상’의 레이블링을 통해 캡슐내시경 영상 학습에 방해가 되는 영상을 분류하는 기법을 제안하였다. 제안하는 기법은 군집화를 통해 불균형하게 분포하고 있었던 다중 ‘학습 노이즈 영상’의 레이블을 균등하게 바꾸어 학습을 수행함으로써, 캡슐내시경에서 발견 가능한 대부분의 ‘학습 노이즈 영상’을 통합하여 제거할 수 있는 효과를 가져온다. 실제 임상 데이터로 실험한 결과, 0.94의 정확도와 0.93의 F1-점수를 보였으며, 데이터 수가 적어 학습될 수 없었던 ‘학습 노이즈 영상’까지 모두 분류될 수 있음을 확인하였다. 향후에는 제안하는 기법을 기반으로 정제된 캡슐내시경 영상을 활용하여, 진단을 보조하기 위한 영상 분류기를 설계할 계획이다.

참고 문헌

[1] Derrington, Dolores. "Artificial intelligence for health and health care." *See https://www. healthit. gov/sites/default/files/jsr-17-task-002_aiforhealthandhealthcare12122017.ddf (last checked 8 November 2018)* (2017).

[2] Litjens, Geert, et al. "A survey on deep learning in medical image analysis." *Medical image analysis* 42 (2017): 60-88.

[3] H. G. Lee, et al. " Intelligent Diagnosis Assistant System of Capsule Endoscopy Video Through Analysis of Video Frame." *Journal of Intelligence and Information Systems* 15.2 (2009): 33-48.

[4] Zhang, Ruikai, et al. "Polyp Detection during Colonoscopy using a Regression-based Convolutional Neural Network with a Tracker." *Pattern Recognition* (2018).

[5] Tarik, Ghannam, et al. "Polyps's region of interest detection in colonoscopy images by using clustering segmentation and region growing." *Information Science and Technology (CiSt), 2016 4th IEEE International Colloquium on*. IEEE, 2016.

[6] Ghosh, Tonmoy, et al. "Cluster based statistical feature extraction method for automatic bleeding detection in wireless capsule endoscopy video." *Computers in biology and medicine*94 (2018): 41-54.

[7] Vilariño, Fernando, et al. "Automatic detection of intestinal juices in wireless capsule video endoscopy." *null*. IEEE, 2006.

[8] Pietri, Olivia, et al. "Development and validation of an automated algorithm to evaluate the abundance of bubbles in small bowel capsule endoscopy." *Endoscopy international open* 6.04 (2018): E462-E469.

[9] D.Y. Yoo, et al. " SVM-based Classification of Over-residue Images for filtering Learning-obstruction Images of Capsule Endoscopy. " *Korea Software Congress (2018): 1865-1867*.

[10] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.