제29권 제1호(통권105호) 2020년 3월 ISSN 1738-6322

소프트웨어공학소사이어티 논문지

Journal of Software Engineering Society

VOLUME 29, NUMBER 1, March 2020

한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

SOFTWARE ENGINEERING SOCIETY

가상화 기술을 활용한 무기체계 소프트웨어 규격자료 품질향상 방안 연구†

(An improvement method of weapon system software standards material quality using virtualization technology)

최 민 관 § 국 승 학 § 이 태 호 §
(Minkwan Choi) (Seunghak Kook) (Taeho Lee)

요 약 최근 무기체계에서 소프트웨어가 차지하는 비중이 증가함에 따라 소프트웨어 개발환경 또한 매우 다양해지고 있다. 무기체계 소프트웨어 분야에서는 국방 규격자료로 소프트웨어 기술문서, 소프트웨어 소스코드, 소프트웨어 실행파일을 제출하도록 하고 있다. 국방 규격을 통해 소프트웨어 실행파일을 재생성하기 위한 소프트웨어 파일목록 및 개발환경을 문서화하도록 요구하고 있다. 하지만 연구개발 종료 후 해당 규격자료를 기반으로 소프트웨어 실행파일을 생성하기 위해서는 기술문서에 작성된 개발환경 정보를 참고하여 개발환경 재구축 등의 추가적인 노력이 필요하다. 따라서 본 연구에서는 가상화 기술을 활용하여 소프트웨어 규격자료의 품질을 향상하는 방안을 제시하고자 한다. 이를 통해 소프트웨어 개발환경 재구축에 대한 노력 절감 및 개발환경 단종으로 인한 문제를 해결할 수 있을 것으로 기대한다.

키워드 : 무기체계 소프트웨어, 국방 규격자료 품질, 가상화

Abstract Recently, software has taken up an increasing share of the weapons system. The software development environment is also becoming very diverse. In the field of weapons software, software technical documents, source codes, and execution files are standardized as defense standards material. Through defense standards, the software file lists and development environments for creating software execution files are required to be documented. However, additional efforts to rebuild the software development environment are needed to recreate the software execution file based on defense standards material after the end of R&D. Therefore, in this study, we propose an improvement method for the quality of software standards material using virtualization technology. This is expected to reduce efforts to rebuild the software development environment and solve problems caused by discontinuation of the development environment.

Key words : Weapon System Software, Quality of Defense Standards Material, Virtualization

1. 서 론

국방 연구개발은 타 산업분야와 다르게 규격화 활동을 통해 국방 표준을 지정하고 있다. 국방 표준화(Standardization)는 군수품의 조달·관리 및 유지를 경제적·효율적으로 수행하기 위하여 표준을 설정하여, 이를 활용하는 조직적 행위와 기술적 요구사항을 결정하는 품목지정, 규격제정, 형상관리 등의 지정에 관한 제반활동을 의미한다. 이러한 국방 표준은 군수품의 다양성 감소로 총수명주기비용 절감 및 획득기간 단축, 군수품 상호운용성·호환성·공통성 증진을 통하여 저비용·고효율의 국방자원 운영체제 구축하는데 활용되고 있다. 또한 사업관리, 품질보증, 원가산정, 계약관리 군수자원관리의 기준으로 제공되고 있다. 국방 규격은 군수품의 조달을 위하여 제품 및 용역의 기술적인 요구사항과 요구필요조건의 일치성 여부를 판단하기 위한 기술문서로 국방규격서, 도면, 부품/BOM(Bill of Material)[1], 품질보증요구서(QAR, Quality Assurance Requirement), 소프트웨어 기술문서 등으로 구성된다.

무기체계가 대형화되고 복잡도가 높아짐에 따라 무기체계에서 중요성 및 소프트웨어가 차지하는 비중이 증가하고 있고, 다양한 타겟 하드웨어를 고려한 다양한 개발환경이 적용되고 있다. 또한 소프트웨어의 구성 역시 다양한 상용(COTS, Commercial Off-The-Shelf), 관급(GOTS, Government off-the-shelf), 그리고 공개

† 본 연구는 2020 한국 소프트웨어공학 학술대회에서 우수 산업체 논문으로 선정되어 소프트웨어공학 소사이어티 논문지로 추천을 받았습니다.
§ 비 회 원 : 국방과학연구소 정보화기술실 SW기술팀
{mkchoi, kuk, lth2094}@add.re.kr
논문접수 : 2020년 02월 28일
심사완료 : 2020년 03월 01일

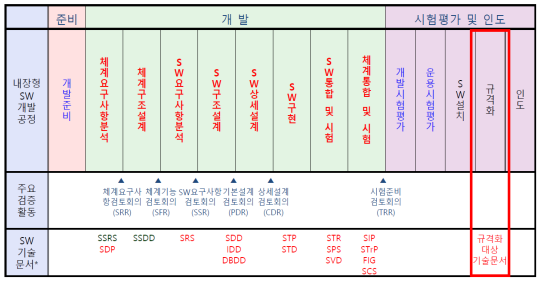
1) BOM: 특정 제품이 어떤 부품들로 구성되는가에 대한 데이터로 최종 제품과 부품관계를 계층적으로 표현한 문서

(Open Source) 소프트웨어를 활용하여 개발되고 있다. 이러한 무기체계 소프트웨어는 개발 완료 이후 소프트웨어의 재생성, 유지보수 등을 위하여 국방 규격을 제정하여 관리하고 있으며 표준화 업무지침, 무기체계 소프트웨어 개발 및 관리 매뉴얼 등의 관련 규정·지침에서 소프트웨어 분야에서 규격화해야 하는 기술자료에 대해 다루고 있다. 하지만 현재 작성되는 소프트웨어 규격자료는 개발이 완료된 소프트웨어 제품에 초점을 맞추고 있고, 소프트웨어 재생성 및 유지보수를 위한 소프트웨어 개발환경 및 개발하는 소프트웨어 이외의 부분에 대해서는 제한적으로 다루고 있다. 따라서 본 연구에서는 무기체계 소프트웨어 규격화 현황에 대해 소개하고 규격화 품질 향상 방안에 대하여 제안하고자 한다.

본 논문의 장절 구성은 다음과 같다. 1장에서는 본 연구에 대한 개략적인 설명을 하였으며 2장에서는 소프트웨어 규격화 현황 및 한계점에 대해 서술한다. 3장에서는 가상화 기술 소개와 가상화 기술의 국방 규격자료에 적용하기 위한 고려사항을 정리하였다. 4장에서는 가상화 기술을 활용한 소프트웨어 규격자료 작성 방안을 제시한다. 마지막으로 5장에서는 결론을 기술한다.

2. 소프트웨어 규격화 현황 및 한계점

그림 1은 무기체계 소프트웨어 개발 단계 별로 주요 검증 활동 및 소프트웨어 기술문서 작성 시기를 요약한 도표이다.

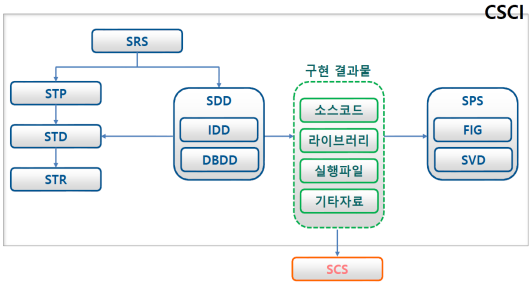


* SDP: SW개발계획서, SRS: SW요구사항명세서, SDD: SW설계기술서, IDD: I/F설계기술서, DBDD: DB설계기술서, STP: SW통합시험계획서, STD: SW통합시험절차서, STR: SW통합시험결과보고서, FIG: 펌웨어설치절차서, SVD: SW버전기술서, SPS: SW산출물명세서, SCS: SW목록명세서, SIP: SW설치계획서

그림 3 무기체계 SW 개발 프로세스

규격화 활동은 개발 및 시험평가가 완료된 이후에 수행한다. 무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 소프트웨어 규격화 시 소프트웨어 개발 프로세스의 각 단계별 작성된 소프트웨어 기술문서, 각종 컴퓨터 파일(소스코드 및 체계의 실행과 관련된 모든 프로그램 파일) 등을 관련 규정·지침(방위사업관리규정, 국방규격·표준서의 서식 및 작성에 관한 지침, 표준화 업무지침)에 따라 표준화 업무 지원 시스템인 국방표준종합정보시스템(KDSIS, Korea Defense Standard Information System)에 제출하도록 요구하고 있다[1-4].

소프트웨어 규격자료는 기술문서, 소프트웨어소스, 그리고 실행파일로 구성되는데 각각의 기술자료 간의 추적성 및 일관성을 가지고 있다. 그림 2는 소프트웨어 규격자료 간의 관계를 표현한 도표이다.



* SRS: SW요구사항명세서, SDD: SW설계기술서, IDD: I/F설계기술서, DBDD: DB설계기술서, STP: SW시험계획서, STD: SW시험절차서, STR: SW시험결과보고서, SPS: SW산출물명세서, FIG: 펌웨어설치절차서, SVD: SW버전기술서, SCS: SW목록명세서

그림 4 SW 규격자료 간 관계

소프트웨어 형상항목(CSCI, Computer Software Configuration Item)의 요구사항이 기술된 소프트웨어 요구사항명세서(SRS, Software Requirement Specification)로 부터 소프트웨어 상세설계가 기술된 소프트웨어설계기술서(SDS, Software Description Specification)와 추적성을 가진다. 시험관련 문서는 SRS, SDD에서 확정된 요구사항 및 설계 내용에 따라 소프트웨어통합시험계획서(STP, Software Test Plan)에 시험 계획을 기술하고, 시험 절차 및 결과를 각각 소프트웨어시험절차서(STD, Software Test Description), 소프트웨어통합시험결과서(STR, Software Test Report)에 기술한다. SDD에 확정된 설계 내용을 바탕으로 구현을 하고 실행파일을 생성한다. 구현 결과물은 소스코드, 라이브러리, 프로젝트 파일을 비롯하여 실행파일 생성에 필요한 모든 파일을 의미한다. 소프트웨어산출물명세서(SPS, Software Product Specification)에는 소스코드, 각종 라이브러리, 실행파일, 기타자료에 대한 파일목록을 기술하고 있다.

국방규격 제정 후 KDSIS에 제출되는 소프트웨어 규격자료의 구성은 그림 3과 같다. 실제 자료는 CSCI 별로 필수 규격화 대상 소프트웨어 기술문서(SRS, SDD, STP, STD, STR, SPS, SIP) 7종 및 소프트웨어소스(소스패키지), 실행파일을 포함하고 있다. 분류체계/속성 자료는 소프트웨어 개발 및 유지보수, 재활용 업무에 사용되는 목록으로 CSCI의 하위 구성요소인 소프트웨어 구성품(CSC, Computer Software Component) 별로 작성하고 있다[5].

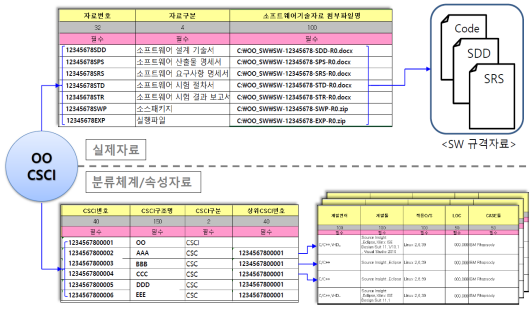


그림 5 국방표준종합정보시스템(KDSIS) 메타데이터 구성

KDSIS는 제출된 소프트웨어 규격자료를 비롯한 모든 규격자료에 대해 이력관리를 하고 있다. 그림 4는 KDSIS 소프트웨어 규격자료 이력관리의 일부 화면을 캡처한 것이다. 소프트웨어 규격자료는 CSCI 별로 자료번호를 부여하고 있으며 국방규격 제정 시 1.0으로 버전을 기록하고, 유지보수 등의 기술변경 시 국방 규격을 개정하고 이때 버전을 2.0, 3.0 순으로 증가하면서 이력관리를 하고 있다.

그림 6 SW 규격자료 이력관리

그러나 현재 관리되는 소프트웨어의 규격자료는 개발이 완료된 소프트웨어 제품에 초점을 맞추고 있어 소프트웨어어의 재생성 및 유지보수를 위한 충분한 정보를 제공하기 어려운 문제가 있다. 그림 5는 SPS, 도면 내 소프트웨어 파일목록 및 개발환경 작성 현황을 도식화한 것이다. 해당 SPS에는 소프트웨어를 구성하는 소스코드, 프로젝트 파일 등 각종 원시 파일목록과 이를 이용하여 실행파일을 생성 및 수정하는 절차를 포함하고 있다. 또한 생성된 실행파일을 타겟 하드웨어에 탑재하는 절차를 포함하고 있다.

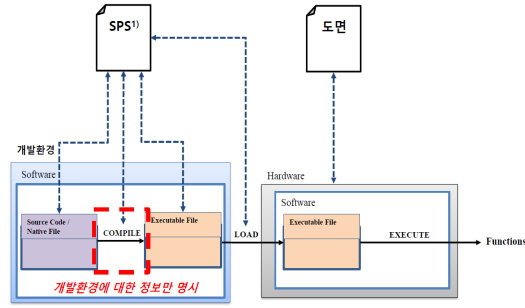


그림 7 SW 규격자료 내 '파일목록 및 개발환경' 작성현황

그러나 SPS에 명시한 모든 파일은 KDSIS에 제출 및 관리되고 있지만 개발환경은 그림 6과 같이 개발 당시에 사용한 개발환경 정보(운영체제, 소프트웨어 개발키트(SDK, Software Development Kit), 컴파일러 등)만 간략히 명시하고 있고, 개발환경 설정 등의 세부 내용은 누락되는 경우가 많다. 실행파일 생성 절차에서 그림 6의 개발환경이 구축된 것을 전제로 절차를 기술하고 있기 때문이다. 따라서 무기체계 소프트웨어 기술변경 등으로 실행파일 재생성이 필요한 경우에는 SPS에 명시된 개발환경 정보를 바탕으로 개발환경을 재구축해야만 한다.

OO SW 개발환경	OO SW 생성절차
(1) 소스코드 개발 운용체제 - windows XP service pack 3.0 (2) 컴파일/빌드 환경 운용체제 - Ubuntu 11.04 (Natty) (3) 소스 코드 개발환경 - source insight 3 v3.50 (4) 컴파일/빌드 환경 - linaro cross-compiler 4.5.2 - python dev 2.7 - automake 11-1 - omniORB 4.2 - omniEvents 2.6.2 - boost library 1.4.4.0 - Qt Embedded library v4.8.2	(1) OO-라이브러리 - OO SW 소스코드를 Linux Host 의 root로 복사한다. - Linux terminal 을 열고, /로 이동한다. - workspace에 대해서 chmod -R 777 권한을 부여 한다. - /workspace로 이동 후, "source CPATH.sh"를 수행한다. - /workspace/cross-workspace/src/△△△-src 로 이동 후, "./reconf.sh && ./CONFIGURE.sh && make install" 를 수행한다. - /workspace/cross-workspace/src/◇◇◇-src로 이동 후, "./reconf.sh && ./CONFIGURE.sh && make install"를 수행한다. - 빌드된 최종 결과물들은 /workspace/cross-workspace/root에 생성된다. (2) ....

그림 8 (SPS) 4. 소프트웨어 생성 및 수정절차 사례

무기체계 소프트웨어는 대부분 임베디드 시스템에 탑재되어 운용되는데 무기체계의 다양성으로 인해 임베디드 소프트웨어 개발환경을 구축하는데 많은 노력이 필요하다. 호환성이 떨어지는 다양한 컴파일러, 민감한 소스코드 빌드 환경, 그리고 보안 이슈로 내부망(Private Network)에서만 운영되는 특성을 갖는다. 또한 타 산업분야와 다르게 연구개발 기간이 길어 주기적인 컴퓨터 하드웨어 업그레이드, 연구개발 당시에 사용하였던 개발환경의 단종, 개발자 퇴사 등의 문제로 기 구축해놓은 개발환경 유지에 어려움을 겪고 있기 때문에 보완이 필요한 상황이다.

3. 가상화 기술 적용을 위한 고려사항

본 논문에서 활용하는 가상화 기술에 대해 소개하고 무기체계(임베디드) 소프트웨어 개발환경에 적용 가능한 가상화 기술을 정리한다. 가상화(Virtualization)라 함은 하나의 물리적 시스템을 논리적으로 분할해 여러 대의 컴퓨터처럼 보이게 하는 소프트웨어 기술을 의미한다. 이러한 가상화 기술은 하이퍼바이저(Hypervisor) 기반과 컨테이너(Container) 기반으로 분류할 수 있다[6-7].

3.1. 하이퍼바이저

하이퍼바이저는 하나의 컴퓨터/시스템에서 여러 개의 운영체제를 가동할 수 있게 하는 가상엔진을 의미하며 가상머신을 통하여 독립된 컴퓨팅 환경을 제공한다. 가상머신(Virtual Machine)은 가상화 대상 리소스인 CPU, RAM, 스토리지, 각종 디바이스 등을 할당 받는다. 하이퍼바이저는 두 가지 유형으로 분류할 수 있는데 Type-1(Native/Bare-metal)은 컴퓨터 하드웨어 상에서 별도의 운영체제의 도움 없이 하이퍼바이저를 직접 동작시키는 방식이다. Type-1 가상화 소프트웨어는 Citrix XenServer, VMware ESXServer 등이 있다. Type-2(Hosted)는 호스트 운영체제에 하이퍼바이저를 설치하고 그 위에서 게스트 운영체제를 동작시키는 방식이다. Type-2 가상화 소프트웨어는 Oracle VirtualBox, VMware Workstation 등이 있다.

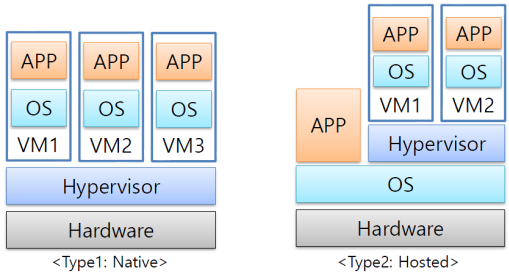


그림 9 하이퍼바이저의 유형[6]

3.2. 컨테이너

컨테이너는 하이퍼바이저와 다소 다른 개념이지만 최근에 클라우드(Cloud) 환경에서 많이 활용되는 기술이다. 그림 8(우측)과 같이 컨테이너는 운영체제 위에서 컨테이너 엔진을 통해 동작하며 어플리케이션과 종속성(Dependency)이 있는 라이브러리, 바이너리, 구성(Configuration)파일, 기타파일 등이 패키지로 묶여서 배포되기 때문에 소프트웨어 실행 오류를 최소화 할 수 있다. 컨테이너 가상화 소프트웨어는 Docker, Kubernetes, AWS ECS 등이 있다.

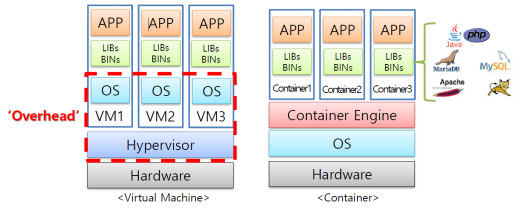


그림 10 하이퍼바이저와 컨테이너 비교[6]

그림 8은 하이퍼바이저 기반과 컨테이너 기반 가상화를 비교한 그림이다. 컨테이너는 운영체제를 포함하고 있지 않기 때문에 가상화 이미지 크기가 가상머신보다 작다. 또한 크기가 작기 때문에 복제와 배포가 용이하다. 운영체제 부팅이 필요하지 않기 때문에 가상화 서비스 실행시간도 상대적으로 빠르다. 각각의 컨테이너는 통제되어 운용되지만 하나의 운영체제를 공유하므로 장애 발생 시 모두 영향을 받는 단점이 있다. 반면에 가상머신은 각각의 가상머신마다 운영체제를 포함하고 있어서 가상화 이미지의 크기가 크고 하드웨어 리소스(CPU, RAM, 스토리지 등)를 많이 요구한다. 각각의 가상머신이 구동하는 게스트 운영체제를 통하여 필요한 하드웨어 리소스를 모두 구동해야하기 때문에 많은 오버헤드가 발생한다. 하지만 컨테이너와 다르게 하이퍼바이저는 가상머신마다 완전히 독립된 공간을 가지고 있기 때문에 안전성, 보안성이 보장된다.

가상화 기술에 대한 소개와 각 기술에 대한 특징을 살펴보았다. 가상화 기술을 무기체계 소프트웨어 개발에 적용하기 위해서 연구개발 및 규격화 측면으로 분리하여 고려가 필요하다. 연구개발 측면에서는 가상화 환경의 성능 및 개발 생산성, 편의성을 고려하여 컨테이너 기반으로 개발환경을 구축하는 것이 많은 이점이 있다. 개발자 간 또는 타 기관과 협업 시 개발환경 공유, 배포가 자유롭고 가상머신에 비해 하드웨어에 의존적이지 않기 때문에 개발 생산성도 높을 것으로 판단된다. 규격화 측면에서는 하이퍼바이저 기반으로 규격자료를 관리하는 것이 필요하다. 연구개발 종료 이후 사용되는 자료이고 규격자료를 형상 관리하는 KDSIS의 자료 관리 방식도 고려해야 한다. 개발 생산성, 편의성 보다 규격자료를 안전하게 보관할 수 있어야 한다. 따라서 연구개발 당시에 구축해 놓은 모든 개발환경(운영체제, SDK, 컴파일러 등)이 가상머신으로 관리하는 것이 타당할 것으로 판단된다.

4. 가상화 기술을 활용한 소프트웨어 규격자료 작성 방안

소프트웨어 개발환경을 가상화 기술을 활용하여 규격화하는 방안을 제시하고자 한다.

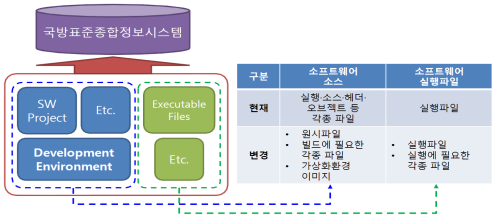


그림 11 가상화 이미지 추가 방법(안)

그림 9는 KDSIS에 제출하는 소프트웨어 규격자료 구성의 기존 방식과 변경(안)을 도식화 한 것이다. 기존 방식은 앞서 설명한 바와 같이 소프트웨어소스 및 기타파일, 실행파일로 구성된다. 변경(안)은 KDSIS 제출하는 소프트웨어 규격자료의 구성을 소프트웨어소스 및 실행파일로 유지하되, 소스코드 빌드 시 필요한 소프트웨어 개발환경(가상화 이미지)를 추가하여 별도로 유지 관리한다.

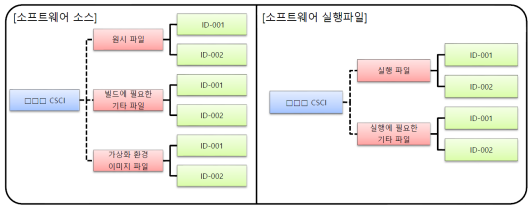


그림 12 SW 규격자료 분류 방법(안)

그림 10은 그림 9(우측)에서 설명한 소프트웨어 규격자료의 분류를 상세화한 것이다. CSCI 내부에 소스코드 프로젝트/빌드 단위는 적게는 하나에서 수십 개까지 다양하게 구성될 수 있다. 동일한 개발환경(운영체제, SDK, 컴파일러 등)에서 개발될 수도 있지만 개발환경이 다양하게 구성될 수 있기 때문에 각 프로젝트/빌드 단위에 따라 가상화 환경 구축이 필요한 상황을 고려해야 한다. SPS 내 디바이스 별로 유일성을 가지는 식별자를 활용(필요시 세부 분류)하여 원시파일과 가상화 이미지 간 추적성을 확보할 수 있게 한다. 또한 그림 10과 같이 소스코드와 가상화 이미지를 구분함으로써 향후 기술변경 시 변경된 부분에 대한 형상관리의 편의성을 향상시킬 수 있다.

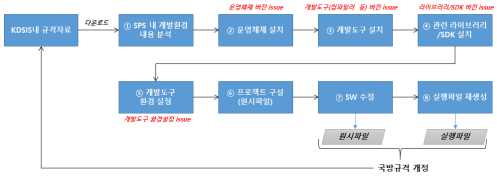


그림 13 SW 규격자료 활용 흐름도: 기존 방식

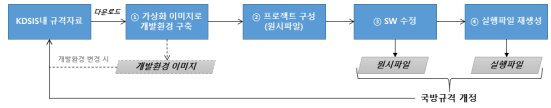


그림 14 SW 규격자료 활용 흐름도: 변경(안)

그림 11, 12는 기존 방식의 소프트웨어 규격자료 활용 과 변경(안)인 가상화 기술을 추가했을 경우를 흐름도로 표현한 것이다. 기존 방식인 그림 11은 KDSIS에서 규격자료를 다운로드 받아 SPS에 기술된 개발환경 내용을 분석한다. 개발환경 구축을 위해 운영체제, 개발도구, 관련 라이브러리/SDK 등을 설치하고 소스코드 빌드를 위해 개발도구, 시스템 환경 설정에 필요한 세부 설정을 한다. 그림 6과 같이 SPS에 텍스트로 기술된 개발정보를 기반으로 새로 구축해야하기 때문에 개발환경 복잡도, 민감도에 따라 해당 과정을 구축하는데 많은 시간을 필요로 한다. 이후 프로젝트를 구성하고 소스코드 수정을 통해 실행파일을 재생성한다. 원시파일, 실행파일을 비롯한 수정된 소프트웨어 규격자료는 국방규격 개정을 통해 반영한다. 그림 12의 변경(안)은 KDSIS에서 받은 규격자료에서 가상화 이미지를 통해 개발환경을 구축한다. 기 구축한 개발환경이 저장된 가상화 이미지를 활용하기 때문에 그림 11의 ①에서 ⑤까지의 작업 및 시행착오를 반복하지 않아도 된다. 이후 절차는 그림 11과 동일하며 개발환경 변경으로 국방규격에 반영이 필요한 경우에만 진행한다.

본 연구에서 제안하는 방안을 실제 소프트웨어 규격자료에 적용하기 위해서는 관련 규정·지침이 보완되어야 한다. 표준화 업무지침(그림 13), 무기체계 소프트웨어 개발 및 관리 매뉴얼(그림 14)에서는 무기체계 소프트웨어 규격화 대상 자료에 대해 다루고 있는데 소프트웨어산출물명세서(SPS)에 포함된 각종 컴퓨터파일을 규격자료로 요구하고 있다. 가상화 이미지는 유지보수 및 재사용을 위한 각종 컴퓨터파일에 해당되므로 추가가 필요하다.

제16조(소프트웨어 규격화) ① 무기체계 소프트웨어의 규격화 대상은 다음 각 호에 따른다. 다만, 규격화 자료를 확보할 수 없는 수입 소프트웨어, 상용 소프트웨어 및 상용 부품에 포함된 소프트웨어는 규격화 대상에서 제외할 수 있다.

..중략..

2 소프트웨어 산출물명세서에 포함된 유지보수 및 재사용을 위한 각종 컴퓨터파일

가. 소스코드

나. 각종 라이브러리

다. 실행파일

라. 개발환경 가상화 이미지

마. 기타자료

그림 15 표준화 업무지침 개정(안)

프로세스	세부 프로세스	산출물	작성 기준	필수 규격화 대상
소프트웨어 구현	단위 소프트웨어 구현 및 데이터베이스 개발	• 소스코드/실행 프로그램 코드(라이브러리/오브젝트코드 포함) • *개발환경 가상화 이미지* • 소프트웨어단위시험결과 • 소프트웨어통합시험계획서 • 소프트웨어통합시험절차서	-	○
	단위 소프트웨어 및 데이터베이스 시험 준비			○
	단위 소프트웨어 시험		-	
	소프트웨어 코드 및 단위시험 결과 검토		CSCI별	○
	소프트웨어 통합시험계획서 갱신 및 시험절차서 개발		CSCI별	○

그림 16 무기체계 SW 개발 및 관리 매뉴얼 개정(안)

가상화 이미지를 통해 바로 개발환경을 구축하고 유지보수 할 수 있다. 또한 개발환경 단종으로 인한 문제를 해결할 수 있다. 기존 방식은 개발환경 단종 시 기존의 개발환경 구축에 필요한 각종 파일 획득 및 재구축에 어려움이 있었지만 가상화 기술을 활용하여 그러한 한계점을 해결할 수 있을 것으로 판단된다.

5. 결 론

본 연구에서는 무기체계 개발에서 수행하는 소프트웨어 규격자료의 품질을 높이기 위해 가상화 기술을 활용하는 방안을 제안하였다. 무기체계 분야에서는 국방 규격자료로 소프트웨어 기술자료를 규격화하도록 요구하고 있지만 해당 기술자료를 기반으로 소프트웨어를 재생성하는데 추가적인 노력이 필요한 상황이다. 본 연구에서는 연구개발 당시의 소프트웨어 개발환경을 가상화 기술을 통해 규격자료에 포함하는 방법을 제시하였다. 연구개발 종료 후 소프트웨어 수정, 실행파일 재생성 등을 위한 개발환경 재구축 노력 및 개발환경 단종으로 인한 문제를 해결할 수 있을 것으로 기대한다.

참 고 문 헌

[1] 방위사업청 예규 제484호, 표준화 업무지침.
[2] 방위사업청, 국방표준업무 실무 가이드북, 2017년 10월.
[3] 방위사업청 예규 제485호, 국방규격·표준서의 서식 및 작성에 관한 지침.
[4] 방위사업청 매뉴얼 제2018-7호, 무기체계 소프트웨어 개발 및 관리 매뉴얼.
[5] 방위사업청, 국방표준종합정보시스템, 2017년 8월.
[6] 안성원, 클라우드 가상화 기술의 변화(SPRi 인사이트리포트 제2018-004호), 2018년 12월.
[7] Compute Virtualization [Online]. Available: https://networklessons.com/cisco/evolving-technologies/compute-virtualization-containers-and-virtual-machines

최 민 관

2011년 용인대학교 컴퓨터정보학부 졸업(학사). 2017년 한양대학교 컴퓨터소프트웨어학과 졸업(석사). 2017년~현재 국방과학연구소 선임연구원. 관심분야는 소프트웨어 공학, 소프트웨어 테스팅.

국 승 학

2004년 충남대학교 컴퓨터과학과 졸업(학사). 2006년 충남대학교 컴퓨터공학과 졸업(석사). 2012년 충남대학교 컴퓨터공학과 졸업(박사). 2012년~현재 국방과학연구소 선임연구원. 관심분야는 소프트웨어 공학, 소프트웨어 테스팅.

이 태 호

1995년 한국과학기술원 화학과/경영정책학과 졸업(학사). 1997년 한국과학기술원 테크노경영대학원 경영공학과 졸업(석사). 2013년 한국과학기술원 정보통신공학과(SW공학전공) 졸업(박사). 1997년~현재 국방과학연구소 SW기술팀 책임연구원/팀장. 관심분야는 소프트웨어 공학, 소프트웨어 품질.

기능 안전 표준 기반의 무기체계 소프트웨어 개발 및 관리 매뉴얼 분석 및 개선 방안 연구†

(Analysis and improvement of weapon system software development and management manual based on functional safety standards)

김 태 현 ‡ 박 다 운 § 백 옥 현 ¶
(Taehyoun Kim) (Daun Bak) (Ockhyun Paek)

요 약 최근 기능 안전에 대한 관심이 높아짐에 따라 다양한 산업 분야에서 기능 안전 표준의 적용이 요구되고 있다. 기능 안전 표준은 시스템의 오작동을 방지하기 위해 필요한 기능 안전 관련 활동들을 정의한 문서이다. 이 표준에 정의된 모든 활동들은 시스템의 위험 분석 및 평가를 통해 산출된 등급 분류 결과에 따라 차등적으로 요구된다. 국내 무기체계 분야에는 방위사업청에서 발간한 무기체계 소프트웨어 개발 및 관리 매뉴얼이 존재한다. 이 매뉴얼은 기능 안전 관련 활동으로 소프트웨어 정적 및 동적 분석 활동을 요구한다. 하지만 해당 매뉴얼에는 선행 활동으로 요구되는 위험 분석 및 평가를 통한 등급 분류 활동 관련 내용이 구체적으로 언급되고 있지 않다. 따라서 본 연구에서는 대표적인 기능 안전 표준들을 기반으로 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 분석하고 이에 대한 개선 방안을 제시하도록 한다.

키워드 : 무기체계 소프트웨어, 기능 안전, 소프트웨어 등급

Abstract As interest in functional safety has recently increased, application of functional safety standards has been required in various industrial fields. A functional safety standard is a document that defines functional safety-related activities required to prevent system malfunctions. All activities defined in this standard are required differentially according to the classification results calculated through the risk analysis and assessment of the system. In the field of domestic weapon systems, there is a manual for the development and management of weapon system software issued by the Defense Acquisition Program Administration (DAPA). This manual requires static and dynamic analysis of software for functional safety related activities. However, the manual does not specifically address the classification activity through risk analysis and assessment as required for the preceding activities. Therefore, in this study, we analyze the problems of the manual based on the representative functional safety standards, and propose improvement plans.

Key words : weapon system software, functional safety, software level

1. 서 론

국내 무기체계 분야 소프트웨어 개발 담당자들은 방위사업청에서 발간한 무기체계 소프트웨어 개발 및 관리 매뉴얼[1]에 따라 소프트웨어를 개발해야 한다. 이 매뉴얼에는 소프트웨어 개발, 관리, 지원 3가지 측면의 프로세스와 각각의 프로세스별 세부 활동들이 정의되어 있으며 이는 소프트웨어 생명 주기 관련 국제 표준인 ISO/IEC/IEEE 12207 Systems and software engineering -- Software life cycle processes (이하 'ISO/IEC/IEEE 12207')[2]를 기반으로 작성되었다.

최근에는 다양한 산업 분야에서 소프트웨어 생명 주기 관련 표준 외에도 기능 안전에 관한 표준의 적용이 요구되고 있다[3-5]. 기능 안전 표준은 시스템의 오작동을 방지하기 위해 필요한 기능 안전 관련 활동들을 정의한 문서이다. 이 표준에 정의된 모든 활동들은 시스템의 위험 분석 및 평가를 통해 산출된 등급 분류 결과에 따라 차등적으로 요구된다.

국내 무기체계 분야의 경우에는 기능 안전 관련 활동으로 소프트웨어의 정적 및 동적 분석 활동이 소프트웨어 개발 담당자들에게 요구된다[1]. 하지만 무기체계 소프트웨어 개발 및 관리 매뉴얼에는 앞서 언급한 기능 안전 활동들의 수행 수준 결정을 위한 내용이 구체적으로 언급되어 있지 않다. 따라서 본 연구에서는 대표적인 기

† 본 연구는 2020 한국 소프트웨어공학 학술대회에서 우수 산업체 논문으로 선정되어 소프트웨어공학 소사이어티 논문지로 추천을 받았습니다.
‡ 회 원 : 국방과학연구소 정보화기술실 SW기술팀
tae_hyoun@add.re.kr
§ 비 회 원 : 국방과학연구소 정보화기술실 SW기술팀
dwpark90@add.re.kr
¶ 비 회 원 : 국방과학연구소 정보화기술실 SW기술팀
ohpaek@add.re.kr
논문접수 : 2020년 02월 28일
심사완료 : 2020년 03월 01일

능 안전 표준들을 기반으로 등급 분류 관점에서 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 분석하고 이에 대한 개선 방안을 제시하도록 한다.

본 논문의 장절 구성은 다음과 같다. 1장에서는 본 연구에 대한 개략적인 설명을 하였으며 2장에서는 본 연구의 배경 지식이 되는 기능 안전 표준 및 무기체계 소프트웨어 개발 및 관리 매뉴얼의 기능 안전 관련 내용을 간략히 이야기하도록 한다. 3장에서는 기능 안전 표준을 기반으로 분석한 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 이야기하며 4장에서 이 문제점에 대한 개선 방안을 제시하도록 한다. 마지막으로 5장 결론을 통해 본 논문을 마치도록 한다.

2. 배경

기능 안전 표준은 시스템의 오작동을 방지하기 위해 필요한 기능 안전 관련 활동들을 정의한 문서이다. 이와 관련된 대표적인 표준으로는 전자, 전기 분야에 적용되는 IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems[3](이하 'IEC 61508')가 존재한다. IEC 61508을 기반으로 자동차, 철도 등 다양한 산업 분야에서 각 분야에 적합한 별도의 기능 안전 표준을 만들어 적용하고 있다[4,5].

기능 안전 표준에 정의된 모든 활동들은 시스템의 위험 분석 및 평가를 통해 산출된 등급 분류 결과에 따라 차등적으로 요구된다. 아래 표 1은 대표적인 기능 안전 표준인 IEC 61508[3]에 정의되어 있는 안전 무결성 등급(Safety Integrity Level, 이하 'SIL')에 대한 표이다. IEC 61508에서는 SIL에 따라 수행되어야 하는 활동의 수준을 달리한다.

Safety Integrity level (SIL)	Average probability of a dangerous failure on demand of the safety function	Average frequency of a dangerous failure of the safety function
4	≥10^-5 to < 10^-4	≥10^-9 to < 10^-8
3	≥10^-4 to < 10^-3	≥10^-8 to < 10^-7
2	≥10^-3 to < 10^-2	≥10^-7 to < 10^-6
1	≥10^-2 to < 10^-1	≥10^-6 to < 10^-5

표 2 IEC 61508[3]의 SIL 분류 기준

국내 무기체계 분야의 경우에는 기능 안전 관련 활동으로 개발자에게 소프트웨어의 정적 및 동적 분석 활동을 요구한다[1]. 무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 정적 분석 활동으로 MISRA-C:2012[6] 등과 같은 코딩 규칙의 준수 및 CWE(Common Weakness Enumeration) 658, 659, 660[7]에 목록화 되어 있는 취약점 항목의 점검, 함수의 복잡도와 같은 소스코드 메트릭 분석 및 제한값 준수를 요구하고 있으며 동적 분석 활동으로는 요구사항 기반의 구조적 coverage 달성을 요구하고 있다.

무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 동적 분석 활동에 한하여 별도의 등급 분류 기준을 제공한다. 아래 표 2와 3은 매뉴얼에 제시되어 있는 등급 분류 기준과 등급 별 요구되는 동적 분석 활동의 수준을 나타내는 표이다. 소프트웨어 개발 담당자는 아래 기준에 따라 등급 별 요구되는 구조적 coverage에 대한 분석을 수행해야 한다.

영향도 (Severity)	
수준	내용
S1	무시 가능
S2	사소함
S3	심각함
S4	치명적
발생 빈도 (Exposure)	
수준	내용
E1	매우 낮은 확률
E2	낮은 확률
E3	중간 확률
E4	높은 확률
제어 가능성 (Controllability)	
수준	내용
C1	간단히 제어가능
C2	보통의 경우 제어가능
C3	제어하기 어렵거나 불가능

표 3 무기체계 소프트웨어 개발 및 관리 매뉴얼의 등급 분류 기준

영향도 (Severity)	발생 빈도 (Exposure)	제어 가능성(Controllability)		
		C1	C2	C3
S1	E1	S	S	S
	E2	S	S	S
	E3	S	S	S
	E4	S	S	S
S2	E1	S	S	S
	E2	S	S	S
	E3	S	S	S
	E4	S	S	B
S3	E1	S	B	B
	E2	B	B	B
	E3	B	B	B
	E4	B	B	M
S4	E1	B	B	M
	E2	B	M	M
	E3	B	M	M
	E4	M	M	M
* S: Statement coverage, B: Branch coverage, M: MC/DC(Modified condition & decision coverage)				

표 4 무기체계 소프트웨어 개발 및 관리 매뉴얼의 등급 분류 기준 별 요구되는 동적 분석 활동 수준

3. 문제점

3.1 시스템 수준과 소프트웨어 수준의 기능 안전 관련 활동간 연계 미흡

국내 무기체계 분야에서는 방위사업관리규정[8]을 통해 사업 수행 시 시스템 수준의 위험 관리 활동 수행을 요구하고 있다. 이를 위해 방위사업청에서는 SE(System Engineering) 기반 위험관리 가이드북[9]을 발간하여 위험 관리 수행 방법과 절차 등을 제공하고 있다. 하지만 해당 가이드북은 시스템 및 소프트웨어의 기능 안전 관련 활동을 언급하고 있지 않으며 등급 분류에 관해서도 아래 표4와 같이 단순 예시만을 제공하고 있다.

구분		발생 가능성			
		낮음(1)	보통(2)	높음(3)	매우 높음(4)
영향성	낮음(1)	1	2	3	4
	보통(2)	2	4	6	8
	높음(3)	3	6	9	12
	매우 높음(4)	4	8	12	16
* 위험 수준에 따라 관리수준 설정: 실무자(1~4), 팀장(6~9), 부장(12), 본부장(16)					

표 5 SE 기반 위험관리 가이드북[9]의 매트릭스를 활용한 위험수준 관리 예시

아래 표 5는 무기체계 소프트웨어 개발 및 관리 매뉴얼에서 요구하는 정적 및 동적 분석 활동에 대한 등급 분류 기준 적용 여부를 분석한 표이다. 표 5에서 보는 바와 같이 정적 분석 활동의 경우에는 등급 분류 기준이 적용되고 있지 않으며 개발되는 모든 소프트웨어에 대해 동일한 수준의 활동이 요구되고 있다. 동적 분석 활동의 경우에는 표 2, 3과 같이 등급 분류 기준이 적용되고는 있으나 시스템 수준의 위험 관리 활동과 별개로 해당 활동이 요구되고 있다.

기능 안전 관련 활동 구분(대)	기능 안전 관련 활동 구분(중)	등급 분류 기준 적용 여부
정적 분석	코딩 규칙 준수	**등급 미분류** 및 모든 소프트웨어에 대해 동일한 기준의 활동 요구
	취약점 점검	
	소스코드 메트릭 제한값 준수	
동적 분석	요구사항 기반 구조적 coverage 분석	**시스템 수준과 별개로 등급 분류** 및 활동 수행

표 6 무기체계 소프트웨어 개발 및 관리 매뉴얼의 정적 및 동적 분석 활동 별 등급 분류 기준 적용 여부

3.2 등급 분류 활동을 위한 자원 부족

무기체계 분야는 다른 산업 분야와 비교했을 때 다양한 시스템이 개발된다는 특징이 존재한다. 따라서 위험 분석 및 평가를 통한 등급 분류 활동을 수행하기 위해서는 상당히 많은 비용과 시간이 필요하다. 또한 개발하는 시스템의 종류는 다양한 반면 각각의 시스템이 양산되는 수량은 타 산업 분야에 비해 현저히 적다는 특징이 있다. 이로 인해 등급 분류 활동을 위한 기반 데이터가 부족하다는 문제가 존재한다.

아래 표 6은 방위사업청에서 제공하는 무기체계 분류체계[10]의 대분류 항목을 나열한 표이다. 무기체계는 대분류 10종, 중분류 42종, 소분류 132종으로 분류된다. 타 산업 분야의 기능 안전 표준을 적용해보면 자동차 분야 기능 안전 표준 ISO 26262[4]는 기동 무기체계 분야의 일부에만 적용될 수 있으며 항공기 분야 기능 안전 표준 DO-178C[5]는 항공 무기체계의 일부에만 적용될 수 있다.

번호	대분류 항목	번호	대분류 항목
1	지휘통제 · 통신 무기체계	6	화력 무기체계
2	감시 · 정찰 무기체계	7	방호 무기체계
3	기동 무기체계	8	기타 무기체계
4	함정 무기체계	9	비무기체계
5	항공 무기체계	10	국방정보체계

표 7 무기체계 분류체계 - 대분류 항목

3.3 타 기능 안전 표준 대비 높은 수준의 동적 분석 활동 요구

2장에서 기술한 바와 같이 매뉴얼에서는 동적 분석 활동을 위해 별도의 등급 분류 기준을 제시하고 있다. 하지만 매뉴얼에서 요구하는 동적 분석 활동의 수준은 타 산업 분야의 기능 안전 표준[3,4,5]에서 요구하는 동적 분석 활동의 수준보다 상당히 높다. 아래 표 7, 8, 9는 대표적인 소프트웨어 기능 안전 표준인 ISO 26262-6[4]과 DO-178C[5]에서 요구되는 동적 분석 활동 수준을 무기체계 소프트웨어 개발 및 관리 매뉴얼의 등급 분류 기준에 대입한 결과를 나타내는 표이다. 표 7, 8, 9에서 보는 바와 같이 타 분야의 기능 안전 표준에서는 등급이 낮은 소프트웨어의 경우 동적 분석 활동을 요구하고 있지 않으나 방위사업청의 매뉴얼은 표3에서 보는 바와 같이 등급이 낮은 소프트웨어에 대해서도 최소 statement coverage 달성을 요구하고 있다.

영향도 (Severity)	발생 빈도 (Exposure)	제어 가능성(Controllability)		
		C1	C2	C3
S1	E1	-	-	-
	E2	-	-	-
	E3	-	-	-
	E4	-	-	-
S2	E1	-	-	-
	E2	-	-	-
	E3	-	-	S
	E4	-	S	B
S3	E1	-	-	-
	E2	-	-	S
	E3	-	S	B
	E4	S	B	B
S4	E1	-	-	S
	E2	-	S	B
	E3	S	B	B
	E4	B	B	M
* -: 활동 미요구 S: Statement coverage, B: Branch coverage, M: MC/DC(Modified condition & decision coverage)				

표 8 ISO 26262의 등급 분류 기준('++' Highly Recommendation 기준) 요구되는 동적 분석 활동을 무기체계 소프트웨어 개발 및 관리 매뉴얼의 등급 분류 기준에 대입한 결과

영향도 (Severity)	발생 빈도 (Exposure)	제어 가능성(Controllability)		
		C1	C2	C3
S1	E1	-	-	-
	E2	-	-	-
	E3	-	-	-
	E4	-	-	-
S2	E1	-	-	-
	E2	-	-	-
	E3	-	-	-
	E4	-	-	-
S3	E1	S	S	S
	E2	S	S	S
	E3	B	B	B
	E4	B	B	B
S4	E1	M	M	M
	E2	M	M	M
	E3	M	M	M
	E4	M	M	M
* -: 활동 미요구 S: Statement coverage, B: Branch coverage, M: MC/DC(Modified condition & decision coverage)				

표 10 DO-178C의 등급 분류 기준 요구되는 동적 분석 활동을 무기체계 소프트웨어 개발 및 관리 매뉴얼의 등급 분류 기준에 대입한 결과

영향도 (Severity)	발생 빈도 (Exposure)	제어 가능성(Controllability)		
		C1	C2	C3
S1	E1	-	-	-
	E2	-	-	-
	E3	-	-	-
	E4	-	-	-
S2	E1	-	-	-
	E2	-	-	-
	E3	-	-	M
	E4	-	M	M
S3	E1	-	-	-
	E2	-	-	M
	E3	-	M	M
	E4	M	M	M
S4	E1	-	-	M
	E2	-	M	B
	E3	M	M	M
	E4	M	M	M
* -: 활동 미요구 S: Statement coverage, B: Branch coverage, M: MC/DC(Modified condition & decision coverage)				

표 9 ISO 26262-6의 등급 분류 기준('+' Recommendation 기준) 요구되는 동적 분석 활동을 무기체계 소프트웨어 개발 및 관리 매뉴얼의 등급 분류 기준에 대입한 결과

4. 개선 방안

4.1 시스템과 소프트웨어 수준의 기능 안전 관련 활동 연계를 위한 제도 개선

3.1절에서 기술한 바와 같이 국내 무기체계 분야에서는 시스템 수준에서 수행되는 기능 안전 관련 활동이 소프트웨어 수준의 활동과 연계되지 않고 있다는 문제점이 있다. 이러한 문제를 해결하기 위해서는 먼저 시스템 수준의 관련 규정과 소프트웨어 개발 및 관리 매뉴얼 간의 연계성 확보를 위한 제도 개선이 필요하다. 정적 분석 활동의 경우 시스템 및 소프트웨어 수준의 등급 분류 기준에 따라 활동의 수준을 달리할 수 있도록 제도 개선이 필요하며 동적 분석 활동의 경우 시스템 수준의 등급 분류 기준과 소프트웨어 수준의 등급 분류 기준이 연계될 수 있도록 제도 개선이 되어야 한다.

본 연구에서는 이를 위해 MIL-STD-882E[11]의 시스템 및 소프트웨어 위험 등급 분류 방법을 국내 무기체계 분야에 적용하는 방안을 제시한다. MIL-STD-882E[11]는 미 국방부에서 발간한 시스템 안전 관련 문서이다. SE 기반 위험관리 가이드북에서 참고하고 있는 미 국방부 위험 관리 관련 문서[12]의 경우 시스템 안전과 관련된 내용에 대해서는 MIL-STD-882E를 참고하도록 하고 있으며 MIL-STD-882E에는 시스템 수준에서부터 소프트웨어 수준까지 연계되는 위험 분석 및 등급 분류 방법이 제시되어 있다. 아래 표 10은 MIL-STD-882E에서 제시하고 있는 시스템의 위험 평가 매트릭스이다. MIL-STD-882E는 시스템의 위험 분석 및 평가 활동과

연계하여 시스템의 위험에 소프트웨어가 미치는 영향의 등급을 결정하기 위한 별도의 기준을 제시하고 있으며 이에 대한 내용은 표 11과 12에서 보는 바와 같다.

Severity / Probability	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

표 11 MIL-STD-882E의 시스템 수준 Risk assessment matrix

Severity / SW Control	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Autonomous (AT)	SwCI 1	SwCI 1	SwCI 3	SwCI 4
Semi-Autonomous (SAT)	SwCI 1	SwCI 2	SwCI 3	SwCI 4
Redundant Fault Tolerant (RFT)	SwCI 2	SwCI 3	SwCI 4	SwCI 4
Influential	SwCI 3	SwCI 4	SwCI 4	SwCI 4
No Safety Impact (NSI)	SwCI 5	SwCI 5	SwCI 5	SwCI 5

* SwCI: Software Criticality Index

표 12 MIL-STD-882E의 Software safety criticality matrix

SwCI	Risk Level
SwCI 1	High
SwCI 1	Serious
SwCI 2	Medium
SwCI 3	Low
SwCI 5	Not Safety

표 13 MIL-STD-882E의 SwCI와 Risk Level 간의 관계

4.2 분류체계 기반 등급 사전 분류 및 정보 제공

3.2절에서 언급한 바와 같이 무기체계 분야는 다른 산업 분야와 비교했을 때 개발하는 시스템의 종류는 다양한 반면 각각의 시스템이 양산되는 수량은 적다는 특징이 존재한다. 이로 인해 위험 분석을 수행하기 위한 기반 데이터가 부족할 뿐만 아니라 상당히 많은 비용과 시간이 필요하다는 문제가 존재한다. 본 연구에서는 이를 해결하기 위해 방위사업청에서 제공하는 무기체계 분류체계[10] 및 국방과학기술 표준분류체계[10]를 기반으로 시스템 및 하위 구성품에 대한 등급을 사전에 분류하고 이에 대한 정보를 제공해주는 방안을 제안한다.

표 4에서 보는 바와 같이 무기체계 분류체계는 시스템 유형을 기준으로 무기체계를 분류하고 있다. 또한 표 10에서 보는 바와 같이 국방과학기술 표준분류체계는 시스템이 아닌 기술적인 측면에서 분류 기준을 제시하고 있다. 이 두 가지 분류체계 내 각각의 항목별로 대표적인 위험 요소를 사전에 분석하고 등급 분류 결과를 시스템 및 소프트웨어 개발 담당자들에게 제공한다면 기반 데이터 및 비용, 시간 부족으로 인한 담당자들의 어려움을 해소할 수 있을 뿐만 아니라 기능 안전과 관련된 시스템 수준 활동과 소프트웨어 수준 활동 간의 연계도 가능할 것으로 기대한다.

번호	대분류 항목	번호	대분류 항목
1	센서	5	추진
2	정보통신	6	화생방
3	제어전자	7	소재
4	탄약/에너지	8	플랫폼/구조

표 14 국방과학기술 표준분류체계 - 대분류 항목

4.3 타 기능 안전 표준과 유사한 수준의 동적 분석 활동 요구

3.3절에서 언급한 바와 같이 매뉴얼에 존재하는 동적 분석 활동의 수준은 다른 산업 분야의 기능 안전 표준[3-5]에서 요구하는 동적 분석 활동의 수준보다 상당히 높다. 만일 개발되는 모든 소프트웨어에 대해 요구사항 기반의 동적 분석 활동을 수행한다면 상당히 많은 시간과 비용이 요구될 것이다. 따라서 투입 노력 대비 효과를 고려하여 타 기능 안전 표준과 유사한 수준의 활동으로 수준을 조정할 필요성이 존재한다.

기능 안전 표준은 동적 분석 활동뿐만 아니라 소프트웨어 생명 주기 각 단계별로 기능 안전을 위한 다양한 활동들을 제시하고 있다. 그리고 이러한 활동들은 위험 분석 및 평가 활동을 통한 등급 분류를 기준으로 수준을 달리하도록 요구된다. 소프트웨어의 기능 안전을 고려한다면 단순히 높은 수준의 동적 분석 활동만을 수행하기보다는 생명 주기 별 여러 활동들을 등급에 따라 수준에 맞게 복합적으로 수행하는 것이 더욱 도움 될 것으로 판단한다.

5. 결론

본 연구에서는 기능 안전 표준을 기반으로 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 분석하였으며 이에 대한 개선 방안을 제시하였다. 국내 무기체계 분야에서는 무기체계 소프트웨어 개발 및 관리 매뉴얼을 통해 기능 안전 관련 활동으로 소프트웨어 정적 및 동적 분석 활동을 요구하고 있다. 하지만 해당 매뉴얼에는 모든 기능 안전 관련 활동의 선행 활동으로 요구되는 위험 분석 및 평가를 통한 등급 분류 활동에 관한 내용이 구체적으로 나와 있지 않다. 본 연구에서는 등급 분류 관점에서 매뉴얼의 문제점을 3가지로 구분하여 분석하였으며 각각의 문제점에 대한 개선 방안을 제시하였다. 이를 통해 무기체계 소프트웨어 분야 개발 담당자들이 기능 안전 관련 활동을 더욱 원활하게 수행할 수 있기를 기대한다.

참 고 문 헌

[1] 방위사업청, “무기체계 소프트웨어 개발 및 관리 매뉴얼”, 2018.

[2] ISO/IEC/IEEE, ISO/IEC/IEEE 12207:2017 Systems and software engineering – Software life cycle processes, 2017.

[3] IEC, IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010.

[4] ISO, ISO 26262 Road vehicles – Functional safety, 2012.

[5] RTCA, DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011.

[6] MISRA, MISRA-C:2012 Guidelines for the use of the C language in critical systems, 2013.

[7] MITRE, https://cwe.mitre.org/

[8] 방위사업청, “방위사업관리규정”, 2019.

[9] 방위사업청, “SE기반 위험관리 가이드북“, 2018.

[10] 방위사업청, 국방과학기술 정보관리 업무지침, 2018.

[11] DOD, MIL-STD-882E Standard practice for system safety, 2012.

[12] DOD, Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs, 2017.

김 태 현

2012년 아주대학교 정보 및 컴퓨터 공학부 졸업(학사). 2014년 한국과학기술원 전산학과 졸업(석사). 2014년~현재 국방과학연구소 연구원. 관심분야는 소프트웨어 공학, 소프트웨어 신뢰성 공학.

박 다 운

2013년 성균관대학교 시스템경영공학과 졸업(학사), 2015년 한국과학기술원 경영공학과 졸업(석사), 2015년~현재 국방과학연구소 연구원. 관심분야는 소프트웨어 신뢰성 공학.

백 옥 현

2000년 충북대학교 정치외교학과(학사). 2002년 충북대학교 전산학과(석사). 2002년~현재 국방과학연구소 연구원. 관심분야는 소프트웨어 아키텍처, 소프트웨어 테스팅.

CNN 모델 평가를 위한 이미지 데이터 증강 도구 개발†

(Development of an Image Data Augmentation Apparatus to Evaluate CNN Model)

최 영 원 ‡ 이 영 우 § 채 흥 석 ¶
(Youngwon Choi) (Youngwoo Lee) (Heung-Seok Chae)

요 약 CNN 모델이 이미지 분류와 객체 탐지 등 여러 분야에 활용됨에 따라, 자율주행자동차와 같이 안전필수시스템에 사용되는 CNN 모델의 성능은 신뢰할 수 있어야 한다. 이에 CNN 모델이 다양한 환경에서도 성능을 유지하는지 평가하기 위해 배경을 변경한 이미지를 생성하는 이미지 데이터 증강 도구를 개발한다. 이미지 데이터 증강 도구에 객체가 존재하는 이미지를 입력하면, 해당 이미지로부터 객체 이미지를 추출한 후 수집한 배경 이미지 내에 객체 이미지를 합성하여 새로운 이미지를 생성한다. CNN 모델 성능 평가 방법으로 개발한 도구를 사용하여 기존 테스트 이미지로부터 새로운 테스트 이미지를 생성하고, 생성한 새로운 테스트 이미지로 CNN 모델을 평가한다. 사례 연구로 Pascal VOC2007 테스트 데이터로부터 새로운 테스트 이미지를 생성하고, 새로운 테스트 이미지로 YOLOv3 모델을 평가하였다. 그 결과 기존 테스트 이미지의 mAP보다 새로운 테스트 이미지의 mAP가 약 0.11 더 낮아지는 것을 확인하였다.

키워드 : 메타몰픽 테스팅, 데이터 어그멘테이션, 합성곱 신경망, 이미지 합성

Abstract As CNN model is applied to various domains such as image classification and object detection, the performance of CNN model which is used to safety critical system like autonomous vehicles should be reliable. To evaluate that CNN model can sustain the performance in various environments, we developed an image data augmentation apparatus which generates images that is changed background. If an image which contains object is entered into the apparatus, it extracts an object image from the entered image and generates composed images by synthesizing the object image with collected background images. As a method to evaluate a CNN model, the apparatus generates new test images from original test images, and we evaluate the CNN model by the new test image. As a case study, we generated new test images from Pascal VOC2007 and evaluated a YOLOv3 model with the new images. As a result, it was detected that mAP of new test images is almost 0.11 lower than mAP of the original test images.

Key words : Metamorphic Testing, Data Augmentation, CNN, Image Composition

1. 서 론

Convolutional Neural Network(CNN)은 이미지 분류와 객체 탐지에 널리 사용되는 모델이다[1]. CNN을 이용한 이미지 인식 알고리즘은 2012년에 AlexNet을 시작으로 계속 발전하였다. AlexNet이 2012년 ImageNet Large Scale Visual Recognition Competition(ILSVRC)에서 에러율 15.3%를 달성하였으며[2], 이후 2014년에 GoogLeNet이 에러율 6.67%를[3], 2015년에 ResNet이 에러율 3.57%를 달성하였다[4].

객체 탐지는 이미지나 영상 속 객체를 탐지하여 해당 객체를 분류한 Label과 Bounding Box를 출력하는 것이다. 객체 탐지 알고리즘도 이미지 인식 알고리즘처럼 CNN을 활용하여 문제를 해결하고자 하였다[5]. 이후 R-CNN[6]을 비롯한 객체 탐지 모델이 등장하였고 자동차의 자율주행 기능에 활용되고 있다. 대표적으로 Google의 Waymo가 자율주행택시를 상용 서비스하고 있다. 그러나 Waymo의 무인 자율주행택시는 법적으로 애리조나 주 피닉스 지역에 한하여 서비스되고 있다. 이는 자율주행자동차에 사용되는 CNN 모델의 학습/테스트를 위한 이미지 데이터가 여러 환경을 반영하지 않는 것으로 추정한다. 인공신경망 모델을 학습하고 성능을 평가함에 있어 학습 데이터와 테스트 데이터가 영향을 미친다. 데이터 셋의 데이터 수가 많더라도 특정 Label 데이터가 높

† 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1F1A1063336).
‡ 비 회 원 : 부산대학교 전기전자컴퓨터공학과
fenrir94@pusan.ac.kr
§ 학생회원 : 부산대학교 전기전자컴퓨터공학과
amorepooh@pusan.ac.kr
¶ 종신회원 : 부산대학교 전기컴퓨터공학부 교수
hschae@pusan.ac.kr
논문접수 : 2020년 02월 28일
심사완료 : 2020년 03월 01일

은 비율을 차지하거나 낮은 비율을 차지하는 데이터 셋으로 모델을 학습시킬 경우에는 Overfitting이 발생할 수 있으며, 해당 데이터 셋을 테스트에 사용하는 경우에는 특정 Label에 대한 테스트만 많이 수행하고 그 이외 Label에 대해서는 테스트를 적게 수행함에 따라 모델을 테스트한 결과를 신뢰할 수 없다. 테스트 데이터의 양이 많더라도 현실에서 발생하는 상황을 반영하지 못한다면, 테스트 정확도가 높게 측정되었더라도 현실의 데이터에 대한 정확도는 낮게 측정된다. 이러한 문제는 자율주행자동차의 CNN 모델에서도 발생할 수 있으므로, 여러 환경에 대하여 CNN 모델을 테스트해야한다.

여러 환경에 대하여 자율주행자동차의 CNN 모델을 테스트하는 방법으로 여러 지역에서 직접 자동차를 운행하면서 테스트하는 방법과 CARLA[7]같은 도구를 사용하여 3D 맵을 제작하고 시뮬레이션 환경을 구축하여 테스트하는 방법이 있다. 그러나 여러 지역을 직접 자동차를 운행하는 방식은 테스트 비용이 크며, 3D 맵을 제작하여 테스트하는 방법은 맵을 제작하기 위한 비용이 필요하다.

본 연구에서는 여러 환경에 대한 CNN 모델의 테스트 비용을 감소시키기 위해 배경을 변경하는 이미지 데이터 증강 도구를 개발한다. 이미지 데이터 증강 도구는 객체가 존재하는 이미지를 입력하면 객체의 형태는 유지하면서 배경을 변경한 이미지를 출력한다. 이미지 데이터 증강 도구에 대한 자세한 설명은 3장에서 다룰 것이며, 개발한 도구를 사용한 사례 연구는 4장에서 설명할 것이다.

2. 배경 지식

2.1. Metamorphic Testing

Test Oracle은 Testing 과정에서 입력에 따른 프로그램 실행 결과가 올바른지 판단하는 절차이다[8]. 복잡한 수식을 계산하는 프로그램이나 컴파일러 프로그램과 같이 입력에 따른 예상 출력을 계산하기 어려워 Test Oracle을 수행할 수 없는 문제를 Oracle Problem이라고 한다[8]. Oracle Problem이 있는 경우 테스트 케이스를 생성하기가 어렵고, 그에 따라 프로그램을 테스트하기 어려운 문제가 있다. 이러한 Oracle Problem을 완화하기 위해 고안된 기법이 Metamorphic Testing이다[9].

Metamorphic Testing은 여러 입력에 따른 프로그램 출력을 통해 예상되는 관계로 테스트 케이스를 생성하는 방법이다. 예를 들어, sin 함수에 12를 입력한다고 할 때 예상 출력을 결정하기 어렵다. 이런 상황에서 sin 함수의 특성이 $\sin(x) = \sin(\pi - x)$이므로, sin(12)의 출력을 정확하게 알지 않아도 식 $\sin(12) = \sin(\pi - 12)$로 테스트할 수 있다. 이러한 프로그램 출력을 통해 추론되는 관계를 Metamorphic Relation이라 한다[9].

본 연구에서는 테스트 이미지 수를 증가시키는 방법으로 합성 이미지 생성과 Data Augmentation 기법을 적용한다. 예를 들어, 테스트 이미지에 개가 들어있을 경우, 개의 이미지를 추출하고, 추출한 개 이미지를 배경 이미지와 합성하여 합성 이미지를 생성한다. 생성된 합성 이미지에는 개가 존재하므로 이미지 분류 또는 객체 탐지 모델은 합성 이미지의 Label을 테스트 이미지의 Label과 동일하게 개로 판별할 수 있어야 한다. 또한 합성 이미지에 Data Augmentation 기법을 적용해도 합성 이미지 내 객체의 형태는 그대로 유지된다. 이에 CNN 모델의 Metamorphic Relation으로 식 테스트 이미지의 Label과 합성 이미지의 Label이 같음을 적용한다.

2.2. 합성 이미지 생성

합성 이미지 생성은 CNN 모델을 평가하기 위해 테스트 이미지로부터 객체 이미지를 추출하고, 추출한 객체 이미지를 배경 이미지와 합성하여 새로운 테스트 이미지를 생성하는 방법이다. 합성 이미지 생성은 객체 이미지 추출과 이미지 합성 두 단계로 이루어진다. 이미지 추출은 객체가 존재하는 이미지에서 객체만 객체 이미지로 추출하는 방법이다. 이미지 합성은 객체 이미지와 수집한 배경 이미지를 합성하고 Data Augmentation 기법을 적용하여 새로운 테스트 이미지를 생성하는 방법이다.

2.2.1. 이미지 추출

이미지 추출은 객체가 존재하는 이미지에서 배경이 없는 객체 이미지를 추출하는 방법이다. 이미지 추출 방법으로 Canny Edge Detection과 GrabCut 알고리즘, Object Detection과 Instance Segmentation이 있다.

Canny Edge Detection은 이미지 내 Edge를 탐색하는 알고리즘으로[10], 해상 알고리즘을 사용해서 이미지 내 객체를 추출할 수 있다. 하지만 Canny Edge Detection을 적용할 때 threshold 값에 따라 객체의 Edge가 판별이 안 되는 경우가 있을 수 있으며, 객체의 일부분만 추출될 수 있다는 단점을 가지고 있다.

GrabCut 알고리즘은 이미지에서 전경을 추출하기 위해 개발된 알고리즘으로, 사용자가 객체를 중심으로 직사각형 Bounding Box를 그리면 객체의 이미지를 추출할 수 있다[11]. 또한 객체의 Bounding Box 내 배경 이미지를 완전히 제거하려면 사용자가 배경과 전경을 설정하는 작업이 필요하여 이미지 생성을 자동화하기 위한 이미지 데이터 증강 도구에 적절하지 않다.

Object Detection은 학습된 Label에 따라 이미지 내 객체를 탐지하여 Bounding Box를 출력하는 방법이다. Object Detection 기법을 할 수 있는 인공신경망 모델로는 YOLOv3[12]와 Faster R-CNN[13] 등 여러 모델이 있다. 인공신경망을 사용하기에 학습된 Label에 대해서만 객체를 탐지할 수 있으며, Bounding Box에 따라 이미지 내 객체 이미지를 추출하면 객체 이외 영역에 기존 이미지의 배경 이미지가 남아있다는 단점이 있다.

Instance Segmentation은 이미지 내 객체의 형태에 따른 Mask 정보로 색을 입히는 방법으로, Instance Segmentation 기능을 하는 모델로 Mask R-CNN이 있

대[14]. Mask R-CNN 모델은 Classification, Object Detection, Instance Segmentation 기능을 가지고 있어, 입력된 이미지의 Mask 정보와 Bounding Box 정보를 사용하여 객체의 형태를 유지하면서 배경 이미지를 제거한 객체 이미지를 추출할 수 있다. Object Detection처럼 학습된 데이터 셋의 Label에 대해서만 객체 이미지를 추출 가능하다는 단점이 있다.

본 연구에서는 객체 이미지 추출을 위한 이미지 추출 방법으로 Instance Segmentation을 사용한다. Instance Segmentation은 Mask 정보를 활용해서 자동으로 배경을 제거하면서 객체 이미지를 추출할 수 있다.

2.2.2. 이미지 합성

이미지 합성은 이미지 추출로 추출한 객체 이미지와 배경 이미지를 합성하는 과정이다. 이미지 합성하는 방법으로는 이미지 더하기와 블랜딩(Blending), 이미지 비트 연산을 사용한 합성이 있다.

이미지 더하기는 두 이미지의 각 좌표의 RGB값을 각각 더하는 방법이다. 더하기 연산 방법을 적용하여 RGB 값이 255보다 큰 값을 가지게 되는 경우 해당 RGB 값을 256으로 나눈 나머지 값 또는 255로 변경한다. 따라서 객체 이미지와 배경 이미지의 RGB 값을 더하는 경우 객체 이미지의 형태가 유지되지 않을 수 있다. 객체 이미지의 형태가 크게 달라지는 경우 Metamorphic Relation을 적용할 수 없는 이미지가 생성된다.

블랜딩은 두 이미지의 RGB값마다 일정 비율을 곱하여 더하는 방법이다. 객체 이미지의 합성 비율이 0에서 1 사이의 실수 α라면 배경 이미지의 합성 비율은 1-α이다. 두 이미지의 합성 비율 합이 1이므로 이미지 합성을 하더라도 RGB값이 255를 초과하지 않는다. 그러나 이 방법도 객체 이미지의 형태가 변형될 수 있다.

이미지 비트 연산은 두 이미지를 and나 xor 같은 비트 연산을 적용하여 합성하는 방법이다. 추출한 객체 이미지를 threshold값에 따라 객체 형태의 Mask 정보를 계산하고, Mask 정보를 사용하여 객체 이미지를 배경 이미지와 합성한다. Mask 정보를 사용하여 배경 이미지와 객체 이미지 간에 섞이지 않고 이미지가 합성이 된다. 하지만 객체 이미지에서 객체와 배경의 구분이 명확하지 않다면 객체의 영역 이외 영역도 Mask 정보에 포함될 수 있다.

본 연구에서는 비트 연산을 사용하여 객체 이미지와 배경 이미지를 합성한다. 비트 연산을 사용하면 객체 이미지의 형태를 유지하면서 배경 이미지와 합성이 가능하다. 또한 이미지 추출에서 Instance Segmentation을 적용하여 객체 이미지의 배경을 일정한 값으로 바꾸어 객체와 배경의 구분이 명확하다면, 비트 연산에서 발생하는 Mask 영역 문제를 해결 할 수 있다.

2.3. Data Augmentation

Data Augmentation은 이미지에 변형 기법을 적용하여 변형된 이미지를 생성하는 방법이다. Data Augmentation은 주로 모델 학습 전에 학습 데이터에 여러 Data Augmentation 기법을 적용하여 학습 데이터의 양을 늘려서 모델을 학습시킨다. Data Augmentation을 적용할 때 주로 사용하는 이미지 변형 기법으로 이미지 대칭, 이미지 크기, 이미지 회전, 이미지 밝기 조절 등이 있으며[15], 그림 1은 각 변형 기법에 따른 Data Augmentation 예시로, 한 이미지로부터 4가지 Data Augmentation 기법을 적용하여 4개의 새로운 변형된 이미지를 생성하였다.

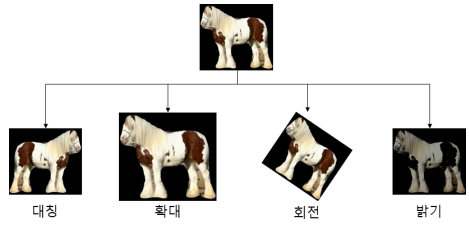


그림 23 Data Augmentation 적용 예시

Data Augmentation을 사용하면 각 테스트 이미지마다 여러 장의 Data Augmentation을 적용한 이미지를 생성할 수 있다. 같은 Data Augmentation 기법을 적용할 때 그 정도에 따라 다른 이미지를 여러 장 생성할 수 있다. 이미지 확대 및 축소의 경우 10%, 20%, 30% 값을 입력하여 크기가 다른 이미지들을 생성할 수 있으며, 회전의 경우 회전 각도를 10°, 20°, 30° 등으로 다르게 적용하여 회전한 정도가 다른 이미지를 생성할 수 있다.

본 연구에서는 테스트 데이터의 양을 증가시키기 위해 Data Augmentation을 적용한다. 평가 대상 CNN 모델은 Metamorphic Relation에 따라 Data Augmentation을 적용한 합성 이미지의 Label와 Data Augmentation을 적용하지 않은 합성 이미지의 Label, 테스트 이미지의 Label은 모두 동일하게 판별해야 한다.

(테스트 이미지의 Label = 합성 이미지의 Label = Data Augmentation을 적용한 합성 이미지 Label)

본 연구에서 적용하는 Data Augmentation 기법들은 아래와 같다. 해당 Data Augmentation 기법들은 imgaug와 scikit-image같은 파이썬 라이브러리에서 지원하는 기법들이다.

- 밝기(Brightness): 이미지의 밝기를 증가 또는 감소함
- 노이즈(Noise): 이미지에 4가지 노이즈 Gaussian Noise, Salt and Pepper Noise, Speckle Noise, Poisson Noise를 적용함
- 날씨(Weather): 비, 안개, 눈 효과를 적용함

3. 이미지 데이터 증강 도구

본 절에서는 기존 테스트 이미지로부터 합성 이미지를 생성하는 도구에 대해서 설명한다. Composed Image Generator는 합성 이미지를 생성하기 위한 도구로, Object Extraction과 Image Composition 2가지 과정으로 이루어져 있다. 그림 2는 Composed Image Generator의 구조와 동작 과정을 나타낸다.

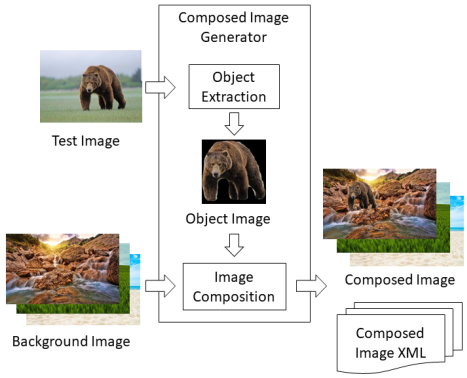


그림 24 Composed Image Generator 동작 과정

Object Extraction 단계에서는 테스트 이미지가 입력되면 테스트 이미지 내에서 객체를 탐지하여 객체 이미지를 추출한다. Image Composition 단계에서는 추출한 객체 이미지와 배경 이미지를 합성하고, Data Augmentation을 적용한다. Composed Image Generator가 동작 완료하면 합성 이미지와 합성 이미지 내 객체 정보를 저장한 XML 파일이 생성된다.

3.1. Object Extraction

Object Extraction은 Instance Segmentation을 적용하여 테스트 이미지를 입력하면 객체를 탐지하여 객체 이미지를 추출한다. 그림 3은 Object Extraction 단계의 동작 과정을 표현한 것이며, Object Extraction 단계는 그림 3의 아래와 같이 3 단계로 동작한다.

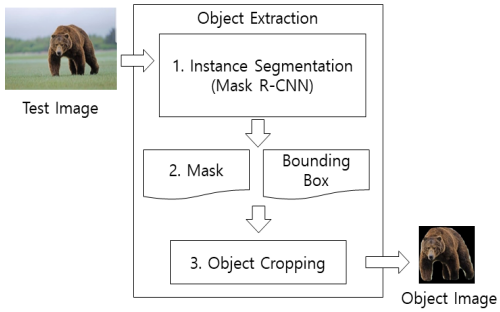


그림 25 Object Extraction 동작 과정

1. 테스트 이미지를 학습된 Mask R-CNN 모델에 입력하여 테스트 이미지 내 객체에 대한 Bounding Box와 Mask 정보를 출력한다.
2. 출력된 Mask 정보를 사용하여 테스트 이미지에서 객체 이외 영역을 배경으로 가정하고 배경 영역을 검은색으로 변경한다.
3. 배경이 검은색으로 변경된 테스트 이미지에서 객체의 Bounding Box 영역을 객체 이미지로 저장한다.

Object Extraction에서는 Instance Segmentation을 적용하기 위해서 MS COCO Data Set[15]을 학습한 Mask R-CNN 모델을 사용한다. MS COCO Data Set의 객체 Label을 80 종류로, 해당 Mask R-CNN 모델은 이미지가 입력되면 해당 이미지 내 학습된 Label의 객체를 탐지하여 해당 객체의 Bounding Box과 Mask 정보를 출력한다. 그림 4는 Object Extraction 단계에서 Instance Segmentation 기능을 하는 Mask R-CNN을 통해 객체 이미지를 추출하는 예시이다. 그림 4-(a)는 Mask R-CNN에 입력한 이미지이며, 그림 4-(b)는 Mask R-CNN 모델이 그림 4-(a) 내 객체를 탐지하여 출력한 해당 객체의 Label, Bounding Box와 Mask를 그림 4-(a)에 적용한 결과이다.

그림 26 Bounding Box와 Mask 적용 예시

그림 5는 그림 4-(a)에서 추출된 객체 이미지이다. 그림 4-(b)에서 Mask 이외 영역의 RGB값을 각각 0으로 변경하면 객체 이외의 영역은 검은색이 된다. 그 후 Bounding Box에 따라 이미지를 잘라서 객체 이미지로 저장하면 그림 5가 생성된다.

그림 27 객체 이미지 예시

3.2. Image Composition

Image Composition은 객체 이미지와 배경 이미지를 입력받아 합성 이미지를 생성한다. 그림 6은 Image Composition 단계의 동작 과정을 표현한 것이며, 세부적인 동작 과정은 그림 6 아래 2 단계로 진행된다.

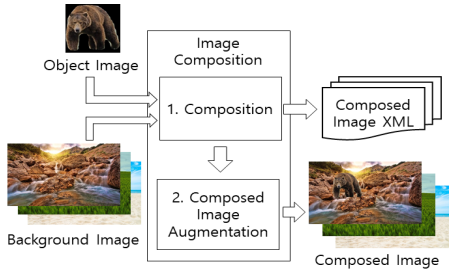


그림 28 Image Compostion 동작 과정

1. 객체 이미지를 비트 연산으로 배경 이미지 내 임의 좌표에 합성하여 저장한다.
2. 합성된 이미지에 설정한 Data Augmentation을 적용한 후 합성 이미지를 저장한다.

Composition 과정에서는 객체 이미지와 배경 이미지를 합성하여 합성 이미지와 합성 이미지 xml을 생성한다. 객체 이미지와 배경 이미지를 합성하기 위해서는 객체 이미지에서 객체의 형태만을 배경 이미지에 합성해야 한다. 이를 위해 객체 이미지를 Grey Scale로 변환한 후 객체의 Mask를 계산하여 해당 객체를 배경 이미지의 임의의 좌표에 합성한다. 객체 이미지의 너비 또는 높이가 배경 이미지보다 큰 경우 배경 이미지와 합성할 수 없으므로 객체 이미지의 너비와 높이를 배경 이미지의 1/3 비율로 축소시킨 후 합성한다.

객체 이미지와 배경 이미지가 합성할 때 객체 이미지의 Label, 너비, 높이, 합성 좌표 정보를 Pascal VOC 포맷에 따라서 XML 파일을 저장한다. 저장한 XML 파일의 정보는 합성 이미지로 모델을 평가하여 출력된 결과와 비교하여 정확도를 측정하기 위해 사용된다.

Composed Image Augmentation 과정에서는 합성된 합성 이미지에 Data Augmentation을 적용한다. 본 연구에서는 합성 이미지에 Data Augmentation 기법으로 Brightness, Noise, Weather를 적용한다. 해당 기법들은 Y. Tian et al.[16]과 M. Koziarski et al.[17]에서처럼 모델 학습 또는 테스트 시 이미지 데이터의 Data Augmentation으로 적용하는 기법들이다. Brightness는 설정한 밝기 값에 따라 이미지의 밝기 값을 변경한다. Noise는 Gaussian Noise와 Salt & Pepper Noise, Poisson Noise, Speckle Noise 총 4가지 노이즈를 이미지에 각각 적용한다. Poisson Noise와 Speckle Noise는 난수 값을 사용한다. Weather는 이미지 합성 방법 중 블랜딩을 사용하여 합성 이미지와 비, 눈, 안개 3가지 날씨별 이미지들을 비는 0.6, 안개는 0.3, 눈은 0.6 비율을 적용하여 합성한다.

4. 사례 연구

본 절에서는 사례 연구로 CNN 모델을 평가하는 과정과 평가 지표, 테스트 데이터, 평가 대상 모델, 모델 평가 결과에 대해 설명한다. 그림 7은 CNN 모델 평가 과정을 나타내며, 아래 3 단계로 진행된다.

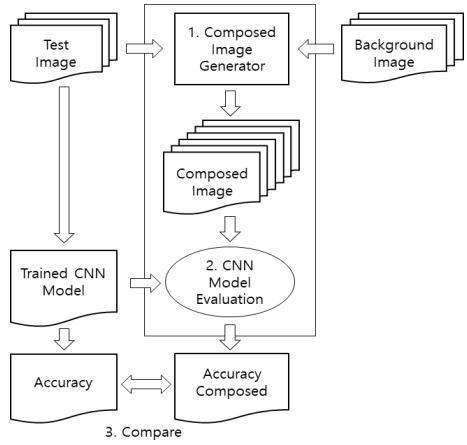


그림 29 CNN 모델 평가 과정

1. Composed Image Generator에 테스트 이미지와 수집한 배경 이미지를 입력하여 합성 이미지를 생성한다.
2. 합성 이미지로 평가 대상 모델의 성능을 측정한다.
3. 합성 이미지로 측정한 모델의 성능과 기존 테스트 이미지로 측정한 모델의 성능을 비교한다.

4.1. 평가 환경

4.1.1 테스트 데이터

테스트 데이터는 Composed Image Generator에서 MS COCO 데이터 셋을 학습한 Mask R-CNN을 사용함에 따라 MS COCO 데이터 셋의 Label을 가지는 데이터 셋을 사용해야 한다. 이에 테스트 이미지로 이미지 인식 대회 Pascal VOC Challenge에서 학습 및 테스트 데이터로 사용된 Pascal VOC 2007[18]의 테스트 데이터 중 Label 10개를 사용한다. 테스트 이미지의 전체 데이터 수는 3707장이며, 이미지 내 객체 크기가 다양하게 분포되어 있다. 이미지의 최대 높이와 너비는 각 500 pixel이다. 표 1은 Pascal VOC2007 데이터 셋의 Label과 Label마다 Test Image 수를 나타내며, 일부 이미지는 Label 객체가 2개 이상 존재한다.

표 1 Pascal VOC2007 데이터 셋 Label 별 이미지 수

Label	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motorbike	Person	Train
이미지 수	239	178	721	322	127	418	274	222	2007	259

4.1.2 평가 대상 모델

평가 대상 모델은 YOLOv3 모델을 사용한다. YOLOv3는 Darknet-53으로 구성된 Object Detection 모델이며, Darknet-53은 Convolutional Layer 53개와 Residual Layer 23개 등으로 이루어져 있다. Darknet-53으로 구성된 YOLOv3는 ResNet-101과 ResNet-152과 비교하여 유사한 성능을 내면서 더 빠른 처리 속도를 보인다[12]. 다른 객체 탐지 모델들보다 빠른 처리 속도로 인해 YOLOv3는 자율주행자동차의 객체 탐지 모델로 연구되고 있다[19, 20]. 평가 대상 모델로 사용하는 YOLOv3 모델은 IoU(Intersection over Union) Threshold를 0.5로 설정하여 Pascal VOC2007과 Pascal VOC2012로 학습된 모델이다.

4.1.3 배경 이미지

배경 이미지는 무료 이미지 사이트 Pixabay에서 Crawling한 후 이미지 분류 Tag에 따라 이미지를 선택한다. 배경 이미지를 선택할 때는 배경 이미지 내 객체가 합성될 객체 이미지의 객체를 탐지하는 것에 장애가 되지 않아야 하므로, 객체가 존재하지 않거나 탐지하기 어려운 객체만 존재하는 이미지를 선택한다. 배경 이미지의 Tag와 그 수는 표 2와 같으며, 총 24장을 선택하였다. 그림 8은 선택한 배경 이미지의 예시다. 배경 이미지의 높이는 340 pixel로 동일하며, 너비는 배경 이미지에 따라 다르며 최소 256 pixel, 최대 1141 pixel이다.

그림 30 Background Image 예시

4.1.4 합성 이미지

합성 이미지 생성에 사용할 테스트 이미지는 테스트 이미지의 10개 Label마다 각각 이미지 20장을 선택하여, 총 200장을 선택한다. 선택한 테스트 이미지 200장과 배경 이미지 24장을 Composed Image Generator에 입력하여 합성 이미지 4,800장을 생성한다.

Composed Image Generator는 생성한 합성 이미지에 Data Augmentation 기법으로 Brightness와 Noise, Weather를 적용한다. Brightness는 밝기를 -100, -50, +50, +100 4가지 밝기 값으로 조절하여 합성 이미지 Brightness를 생성한다. Noise는 Gaussian Noise와 Salt & Pepper Noise, Poisson Noise, Speckle Noise 총 4가지 노이즈를 적용하여 합성 이미지 Noise를 생성한다. Weather 기법은 비, 안개, 눈에 대해 각각 2가지 효과를 적용하여 합성 이미지 Weather를 생성한다. 각 기법을 적용한 합성 이미지의 수는 아래와 같다.

- 합성 이미지 Brightness:
  4,800 images * 4 brightness = 19,600 images
- 합성 이미지 Noise:
  4,800 images * 4 noise = 19,600 images
- 합성 이미지 Weather:
  4,800 images * 3 weather * 2 effects/weather = 28,800 images

그림 9는 합성 이미지와 합성 이미지에 Data Augmentation 기법들을 적용한 이미지의 예시이다. 그림 9-(a)는 배경만 변경된 합성 이미지이고, 그림 9-(b)는 그림 9-(a)에 Brightness -100을 적용한 합성 이미지 Brightness이다. 그림 9-(c)는 그림 9-(a)에 Noise 기법 중 Speckle Noise를 적용한 합성 이미지 Noise이고, 그림 9-(d)는 Weather의 비 효과를 그림 9-(a)에 적용한 합성 이미지 Weather이다.

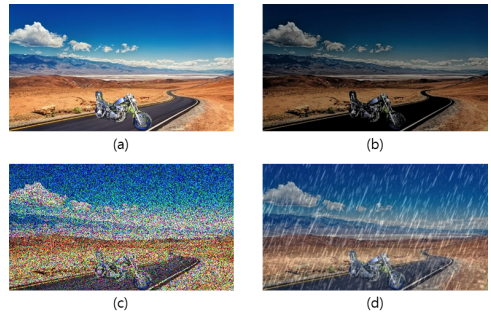


그림 31 합성 이미지 예시

4.1.4 평가 지표

모델의 성능 평가 지표는 Pascal VOC2007 테스트 데이터와의 성능 비교를 위하여 mean Average Precision(mAP)과 각 Label의 Average Precision(AP)을 사용한다. 본 연구에서 개발한 도구에서 생성하는 이미지는 객체가 1개 존재하지만 Pascal VOC2007 테스트 데이터에는 여러 객체가 존재하는 이미지도 있으므로 mAP와 AP를 사용한다. Average Precision은 Label 별 Recall에 대한 Precision의 평균이며, mean Average

표 2 배경 이미지 Tag 별 이미지 수

Tag	Arctic	City	Desert	Field	Forest	Mountain	Road	Village	총합
이미지 수	3	3	4	2	3	3	4	2	24

Precision은 Average Precision의 평균이다.

객체 탐지 판단을 위한 IoU(Intersection over Union) Threshold 값은 평가 대상 모델이 학습할 때와 같이 0.5를 적용한다. IoU Threshold는 그림 10처럼 객체의 Bounding Box와 모델이 예측한 Bounding Box로 계산한다. IoU Threshold가 0.5라는 것은 실제 객체의 Bounding Box와 모델이 예측한 객체의 Bounding Box가 겹치는 영역이 두 Bounding Box를 합친 영역의 1/2 이상인 경우 객체를 탐지했다고 판단하여 객체를 Label에 따라 분류를 하고, 그렇지 않은 경우 객체를 탐지하지 못했다고 판단한다.

그림 32 IoU Threshold 계산식

4.2. 평가 결과

모델 평가는 테스트 이미지와 합성 이미지, 합성 이미지 Brightness, 합성 이미지 Noise, 합성 이미지 Weather로 각각 평가 대상 모델 YOLOv3를 평가하여 출력된 mean Average Precision(mAP)과 각 Label의 Average Precision(AP)으로 분석한다. 표 3은 각 테스트 데이터들을 사용하여 평가 대상 모델인 YOLOv3 모델을 평가한 결과 mAP와 각 Label의 AP를 정리한 표이다.

4.2.1 테스트 데이터 별 mAP 분석

테스트 이미지에 대한 mAP보다 합성 이미지에 대한 mAP가 더 낮게 측정되었다. 합성 이미지 Brightness와 합성 이미지 Noise, 합성 이미지 Weather에 대한 모델의 mAP는 합성 이미지의 mAP보다 더 낮아진 것을 확인할 수 있으며, 합성 이미지 Brightness, 합성 이미지 Weather, 합성 이미지 Noise 순서로 mAP가 작아진다.

이를 통해 평가 대상 모델은 테스트 데이터 중 다른 배경 이미지와 합성되어 생성된 합성 이미지의 mAP가 테스트 데이터의 mAP보다 낮아졌으며, 합성 이미지에 Data Augmentation을 적용한 데이터 셋들의 mAP가 더 많이 낮아지는 것을 확인하였다.

4.2.2 테스트 데이터 별 AP 분석

테스트 이미지의 AP와 비교하여, 합성 이미지에서 Bus와 Cow, Person의 AP는 다른 Label보다 상대적으로 적게 감소하였으며 절대적인 감소 수치도 0.05 미만이다. 반면에 Train의 AP가 약 0.23으로 가장 크게 감소하였고 Bicycle의 AP가 약 0.20으로 두 번째로 많이 감소하였다. Cat과 Dog의 AP 또한 0.16 이상 감소하였다.

이러한 결과를 통해 평가 대상 모델은 Bus와 Cow, Person에서 배경이 변경하여도 성능을 거의 유지할 수 있을 정도로 학습이 잘 되었다고 간주되며, Train과 Bicycle, Cat, Dog에서 배경이 변경된 이미지에 대한 학습이 부족한 것으로 간주된다.

합성 이미지의 AP와 비교하여, 합성 이미지 Brightness와 합성 이미지 Noise, 합성 이미지 Weather의 AP가 더 낮게 측정되었다. 합성 이미지 Brightness의 AP가 다른 두 Data Augmentation을 적용한 데이터 셋들보다 상대적으로 적게 감소하였으며, 합성 이미지 Noise의 AP는 다른 두 Data Augmentation 기법을 적용한 데이터 셋보다 전체적으로 합성 이미지의 AP에서 더 많은 폭으로 AP가 감소하였다. 예외로 Cat의 경우 합성 이미지 Weather에 대해 AP가 가장 많이 감소하였다. Cow와 Person의 경우 합성 이미지 Noise와 합성 이미지 Weather의 mAP가 약 0.02 차이를 보이며 다른 Label에 비해 그 차이가 상대적으로 작다.

이러한 결과를 통해 적용한 Data Augmentation 기법 중 Noise가 Brightness와 Weather보다 더 객체를 식별하기 힘들게 이미지를 변형했다고 추정한다. Noise 중에서도 Speckle Noise가 가장 객체를 식별하기 힘들게 변형시켰다고 판단한다. 또한 Brightness는 Noise와 Weather보다 덜 객체를 변형시켰다고 추정한다. 그러나 본 사례 연구에서는 배경이 변경된 이미지에 Data Augmentation을 적용함에 따라 이미지 변형에 시너지가 발생했을 가능성이 있다. 이는 기존 테스트 이미지에 Data Augmentation 기법만 적용한 이미지로 평가 대상 모델을 측정해야 보다 객관적으로 분석할 수 있다.

표 3 테스트 데이터 별 mAP와 각 Label의 AP

테스트 데이터	mAP	Label(AP)									
		Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motorbike	Person	Train
테스트 이미지	0.930	0.930	0.953	0.943	0.940	0.889	0.928	0.931	0.939	0.914	0.935
합성 이미지	0.816	0.735	0.906	0.861	0.759	0.877	0.760	0.783	0.883	0.901	0.697
합성 이미지 Brightness	0.735	0.686	0.871	0.826	0.673	0.738	0.667	0.685	0.802	0.819	0.585
합성 이미지 Noise	0.645	0.588	0.716	0.746	0.588	0.659	0.581	0.603	0.690	0.754	0.522
합성 이미지 Weather	0.698	0.671	0.850	0.811	0.542	0.676	0.634	0.654	0.793	0.773	0.578

4.2.3 관련 연구와의 비교

본 사례 연구를 통하여 본 연구에서 개발한 이미지 데이터 증강 도구로 생성한 이미지가 기존 테스트 이미지보다 평가 대상 모델의 성능이 낮게 측정되는 것을 확인하였다. 기존 연구와 비교하여, 파이썬 라이브러리 scikit-image와 imgaug는 다양한 Data Augmentation 기법을 제공하지만 이미지의 배경을 다른 배경으로 변경하는 기능은 지원하지 않는다. Alhaija et al.[21]은 AR을 사용하여 여러 배경을 적용한 현실적인 이미지를 생성할 수 있다. 그러나 이미지 생성을 위해 360도 파노라마 이미지가 필요하며 객체 Label이 명시되지 않은 경우 수동으로 이미지 내 객체 Labeling 작업을 수행하므로 Alhaija et al.[21]은 본 연구에서 개발한 도구보다 이미지 생성을 위한 비용이 크다고 추정할 수 있다.

본 연구에서 부족한 점은 생성된 이미지가 올바르게 합성되어 테스트 데이터로 사용가능한 이미지라고 정량적으로 판단할 수 있는 기준이 없다. 본 연구의 도구에서 객체 이미지와 배경 이미지를 합성할 때 객체와 배경 간의 현실성을 고려하지 않고 배경 이미지 내 임의의 좌표에 객체 이미지를 합성한다. 예를 들어 자동차 객체 이미지와 해안 도로 배경 이미지를 합성할 경우 자동차가 바다 위에 존재하는 이미지가 생성될 수 있다. 이러한 비현실적인 이미지들을 직접 보면서 판단하지 않고 자동으로 검출하거나 생성되지 않도록 보완해야한다.

5. 결론 및 향후 연구

본 연구에서는 CNN 모델의 여러 환경에 대한 테스트 비용을 줄이기 위해 새로운 테스트 이미지를 생성하는 도구를 개발하였다. 테스트 데이터와 배경 이미지, 적용하는 Data Augmentation 기법의 설정 값을 이미지 데이터 증강 도구에 입력하면 자동으로 합성 이미지를 생성한다. 사례 연구로 Composed Image Generator를 사용하여 Pascal VOC2007로부터 배경을 변경한 새로운 테스트 데이터를 생성하여 YOLOv3 모델을 평가하였다. 그 결과 기존 테스트 데이터로 모델을 평가한 결과보다 배경을 변경한 테스트 데이터로 모델의 평가한 결과의 mAP와 AP가 전체적으로 더 낮게 나오는 것을 확인하였다.

본 연구에서는 이미지 합성을 할 때 객체 이미지와 배경 이미지 간의 연관성을 고려하지 않고 배경 이미지 내 임의의 좌표에 합성하였다. 이는 비현실적인 합성 이미지가 생성될 수 있어 모델 평가 시 성능 평가 결과를 신뢰할 수 없을 수 있다. 향후 연구에서는 테스트 이미지의 Label과 배경 이미지의 Tag 간 연관관계를 정의한다. 정의한 연관관계에 따라 합성 여부와 객체 이미지를 배경 이미지에 합성하는 좌표를 결정하여 이미지를 생성하여 모델을 평가하는 연구를 수행할 계획이다.

참 고 문 헌

[1] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, “Importance Estimation for Neural Network Pruning,” in *CVPR*. pp. 11264-10272, 2019.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *NIPS*, pp. 1097-1105, 2012.

[3] C. Szegedy. W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” in *CVPR*, pp. 1-9, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *CVPR*, pp. 770-778, 2016.

[5] C. Szegedy, A. Toshev, and D. Erhan, “Deep Neural Networks for Object Detection,” in *NIPS*, pp. 2553-2561, 2013.

[6] C. Szegedy, A. Toshev, and D. Erhan, “Deep Neural Networks for Object Detection,” in *NIPS*, pp. 2553-2561, 2013.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *CoRL*,, pp. 1-16, 2017.

[8] S. Segura, G. Fraser, A.B. Sanchez, and A. Ruiz-Corts, “A Survey on Metamorphic Testing,” *IEEE TSE*, Vol. 42, No. 9, pp. 805-824, 2016.

[9] T. Y. Chen, S. C. Cheung, and S. M. Yiu, “Metamorphic Testing: A New Approach for Generating Next Test Cases,” Computer Science, Hong Kong Univ. of Science and Technology, Tech. HKUST-CS98-01, 1998.

[10] J. Canny, “A Computational Approach to Edge Detection,” *IEEE TPAMI*, Vol. 8, No. 6, pp. 679-698, 1986.

[11] C. Rother, V. Kolmogorov, and A. Blake, “‘Grabcut’: Interactive Foreground Extraction Using Iterated Graph Cuts,” in *ACM TOG*, Vol. 23, No. 3, pp. 309-314, 2004.

[12] J. Redmon, and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv: 1804.02767*, 2018.

[13] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *NIPS*, pp. 91-99, 2015.

[14] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in *ICCV*, pp. 2961-2969, 2017.

[15] T.-Y. Lin, M.Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *ECCV*, pp. 740-755, 2014.

[16] Y. Tian, K. Pei, S. Jana, and B. Ray, “DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars,” in *ICSE*, pp. 303-314, 2018.

[17] M. Koziarski and B. Cyganek, “Image Recognition with Deep Neural Networks in Presence of

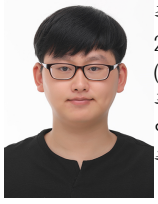
Noise-Dealing with and Taking Advantage of Distortions," *ICAE*, Vol. 24, No. 4, pp. 337-349, 2017.

[18] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *IJCV*, Vol. 88, No. 2, pp. 303-338, 2010.

[19] J. Choi, D. Chun, H. Kim, and H.-j. Lee, "Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving," in *ICCV*, pp. 502-511, 2019

[20] Q. Mao, H. Sun, Y. Liu, and R. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications," *IEEE Access*, Vol. 7, No. 1, pp. 133529-133538, 2019.

[21] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes," *IJCV*, Vol. 126, No. 9, pp. 961-972, 2018



최 영 원

2020년 부산대학교 전기컴퓨터공학부 졸업(학사). 2020년~현재 부산대학교 전기전자컴퓨터공학과 석사과정. 관심분야는 소프트웨어 공학, 소프트웨어 테스팅, 데이터 기반 심층신경망 분석

이 영 우

2016년 부산대학교 정보컴퓨터공학부 졸업(학사). 2018년 부산대학교 전기전자컴퓨터공학과 졸업(석사). 2018년~현재 부산대학교 전지전자컴퓨터공학과 박사과정. 관심분야는 소프트웨어 공학, 데이터 기반 결함 분석

채 홍 석

1994년 서울대학교 원자핵공학과 졸업(학사). 1996년 한국과학기술원 전산학과 졸업(석사). 2000년 한국과학기술원 전산학과 졸업(박사). 2000년~2003년 동양시스템즈 기술연구소 선임연구원. 2003년~2004년 한국과학기술원 전산학과 초빙교수. 2004년~현재 부산대학교 컴퓨터공학과 교수. 관심분야는 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 검증

논문지 논문 모집 (Call for Papers)

한국정보과학회 소프트웨어공학 소사이어티에서는 매년 4회에 걸쳐 '소프트웨어공학 소사이어티 논문지'를 발간하고 있습니다. 이 논문지에는 소프트웨어공학 전반에 걸친 연구 성과 및 산업계 동향을 소개하고 있습니다. 이에 다음과 같은 소프트웨어공학 주제에 관련된 논문을 모집하고 있으니 학계와 산업계의 여러분들의 적극적인 논문투고를 바랍니다.

◈ 논문 주제

○ 소프트웨어 설계, 아키텍처 및 프로덕트라인
○ 요구공학
○ 소프트웨어 품질 및 테스트
○ 프로젝트 관리 및 프로세스
○ 소프트웨어 정형 기법 및 검증
○ 임베디드, 모바일, 웹기반 소프트웨어 개발
○ 기타 소프트웨어 응용 (국방, 자동차, 의료, 조선, IoT, 빅데이터 등의 분야)

◈ 논문심사

○ 투고된 논문은 편집위원회에서 심사 선정하며, 필요 시 외부 심사위원을 위촉하여 심사를 합니다. 제출된 논문은 반환하지 않습니다.
○ 심사료 및 게재료: 없음

◈ 논문 제출

○ 한국정보과학회 논문지 투고 양식(www.kiise.or.kr)을 사용하며, 논문의 분량은 10장으로 제한합니다.
○ 논문지 투고규정에 따라 작성된 심사용 논문파일과 함께 논문제목, 저자, 소속, 초록을 편집위원장 또는 편집위원에게 이메일로 송부하시기 바랍니다.

◈ 문의처 (편집위원회)

○ 편집위원장 : 이선아 교수 (경상대학교, 055-772-1377, saleese@gnu.ac.kr)
○ 편집이사 : 배경민 교수 (POSTECH 054-279-2256, kmbae@postech.ac.kr)
○ 편집이사 : 홍 신 교수 (한동대학교, 054-260-1409, hongshin@handong.edu)

투 고 요 령

1. 소프트웨어공학소사이어티 논문지에 실리는 원고는 주제 논문, 일반 논문, 산업체 기고 등으로 구분하며 다음과 같은 분야에 대하여 모집한다.

   가. 소프트웨어공학 및 그 응용분야에 대한 연구결과
   나. 강좌 및 관련 교육사항 소개 (목적, 과정, 일정, 대상, 특징)
   다. 소프트웨어 도구 및 방법론 소개 (가격, 특징, 종류, 적용사례)
   라. 소프트웨어 산업에 대한 학계, 업계의 주요 관심사
   마. 기타 관련 사항

2. 투고자는 원칙적으로 본 소사이어티의 회원으로 한다. 다만 공동 또는 초청 기고자는 예외로 한다.

3. 논문은 원칙적으로 한글로 작성한다.

4. 원고는 한글(hwp), 워드(MS Word), PDF 형식 중 하나를 택하여 작성하며, 그림과 표를 포함하여 10쪽 이내로 한다.

5. 논문 내용에 직접 관련이 있는 문헌에 대해서는 이들 문헌에 관련이 있는 본문 중에 참고 문헌 번호를 쓰고 그 문헌을 참고문헌 난에 인용 순서대로 기술한다. 참고문헌은 학술지의 경우 저자, 제목, 학술지명, 권, 호, 쪽수, 발행 연도의 순서로, 단행본은 저자, 서명, 쪽수, 발행처, 발행 연도의 순서로 기술한다.

6. 작성된 논문은 논문파일과 함께 논문제목, 저자, 소속, 초록을 편집위원장 또는 특집 주제 담당 편집위원에게 이메일로 제출한다.

7. 기타 자세한 사항은 한국정보과학회 논문지 투고 요령을 따른다.

소프트웨어공학소사이어티 논문지 제29권 제1호 (통권 105호)

발 행 일 ‖ 2020년 3월 31일

발 행 인 ‖ 이병정

편 집 인 ‖ 이선아

발 행 처 ‖ 사단법인 한국정보과학회 소프트웨어공학소사이어티

연 락 처 ‖ 서울특별시 동대문구 서울시립대로 163(전농동) 서울시립대학교
정보기술관 202호 컴퓨터과학부 이병정
전 화 : 02-6490-2451, 팩 스 : 02-6490-2444
홈페이지 : http://www.sigsoft.or.kr/

Journal of Software Engineering Society

VOLUME 29, NUMBER 1, March 2020

Korean Institute of Information Scientists and Engineers

Software Engineering Society