Journal of the KIMST, Vol. 17, No. 2, pp. 213-222, 2014
ISSN 1598-9127
DOI http://dx.doi.org/10.9766/KIMST.2014.17.2.213

Research Paper 정보기술 부문

무기체계 내장형 소프트웨어 신뢰성 향상을 위한 MAAB 스타일 가이드라인 영향성 연구

김연균*,1) · 윤형식1)

1) 국방과학연구소

Research on the Effects of MAAB Style Guidelines for Weapon System Embedded Software Reliability Improvement

Yeon-Gyun Kim*,1) · Hyung-Sik Yoon1)

1) *Agency for Defense Development, Korea*

(Received 28 October 2013 / Revised 27 January 2014 / Accepted 21 February 2014)

ABSTRACT

In this paper, we introduce that MAAB style guideline has effects on the codes generated from Simulink models for static and dynamic software testing, when weapon system embedded software design and implementation are performed using the model based method. As showing the effects, MAAB guideline is helpful for defect prevention related with coding rules and run time errors associated with the DAPA weapon system embedded software guide. Thus, we check related items between MAAB and DAPA software reliability testing including static and dynamic analysis. And then we propose the criterion to select proper items from MAAB for DAPA guideline and show how to verify the relationship and the effects on reliability of models in Simulink. In addition, we show the needs for clear logics in conditional block models or statements and simple complexity models for Simulink model based design.

Key Words : MAAB(Mathworks Automotive Advisory Board), Embedded Software Reliability Testing(내장형 소프트웨어 신뢰성 시험)

1. 서 론

항공기, 자동차, 의료장비 등 높은 신뢰성이 요구되는 안전필수(safety-critical) 시스템에서 주로 실시간 처리를 수행하는 내장형 소프트웨어(이하 SW)가 차지하는 비중은 계속 늘어나고 있다. 특히 항공 구동기 제어 및 항법 등을 담당하는 비행조종컴퓨터(Flight Control Computer)에 탑재된 OFP(Operational Flight Program) 개발 시, 개발 도구가 제공하는 블록 다이어그램, 상태 기계(state machine)에서 제공하는 다양한 편의성 때문에 Simulink, SCADE, LabVIEW 등과 같은

* Corresponding author, E-mail: yg_kim@add.re.kr

모델 기반 SW 개발도구를 사용하는 사례가 증가하고 있다.

군용 항공기에 적용된 모델 기반 내장형 SW는 코드의 신뢰성 보장을 위해 방위사업청(이하 DAPA)의 신뢰성 시험 방법 및 평가 기준을 준수해야 한다[1]. 또한 SW의 신뢰성은 항공기의 안정성과 직접 관련되어 있기 때문에 감항인증에서도 SW 품질보증에 관련된 프로세스 및 산출물을 평가한다. 최근 국제 감항인증 표준인 DO-178B에서 개정된 DO-178C의 DO-331에서는 DO-178B에서 다루지 않았던 모델 기반 개발 및 검증 프로세스를 적용하고 있으며, 모델 설계 가이드라인에 따라 SW를 개발하도록 규정하고 있다[2].

Simulink를 이용하여 설계하는 경우, MAAB와 같은 모델 설계 가이드라인에서 제공하는 모든 항목을 무기체계 내장형 SW 개발에 적용하는 것은 소요 시간과 그 효과를 입증하는데 어려움이 있기 때문에 개발 분야마다 필수적으로 최적화된 가이드라인 항목 선정 및 개발에 대한 연구가 필요하다. 하지만 국내에서는 모델 설계 가이드라인이 모델에서 생성된 소스코드의 신뢰성에 미치는 영향성을 분석하고 실용적인 가이드라인 항목을 선정하는 기준에 대한 연구는 이루지지 않고 있다.

본 논문에서는 군용 항공기에 탑재되는 비행조종컴퓨터 OFP 개발 시, Simulink를 이용한 모델 기반 개발을 적용하기 위하여 MAAB 스타일 가이드라인 항목 중 DAPA에서 제안하는 무기체계 내장형 SW의 신뢰성 시험 방법 및 기준과 연관된 항목을 확인한다. 해당 항목들이 모델에서 자동으로 생성된 코드의 정적시험과 동적시험에 미치는 영향성을 분석하여 DAPA의 SW 신뢰성 시험에 적합한 항목을 선정하는 기준을 제시하고 그 선정된 항목의 영향성을 검증하는 방법을 제안한다.

2. 본 론

2.1 모델기반 SW 신뢰성 시험

2.1.1 신뢰성 시험 프로세스

Mathworks의 Simulink의 경우, 블록 다이어그램, 상태기계, 데이터 사전 등을 이용하여 SW 개발에 필요한 알고리즘을 모델링하고 비교적 간편하게 모의(simulation)를 수행할 수 있다. 개발 도구에서 제공하는 사용자 편의성 때문에 다양한 제어 시스템의 내장형 SW 개발 도구로 사용되고 있다.

Fig. 1은 Simulink에서 제공하는 DO-178B Tool Qualification 개발 도구의 기능과 SW 개발 산출물과의 연관성을 개략적으로 나타낸 것이다. 시스템 요구사항에 따라 Simulink로 모델을 설계하고, Model Advisor를 사용하여 모델에 대한 표준 적합성 여부를 확인한다. 또한 Simulink에서는 타겟 프로세서의 조건에 따라 Embedded Coder를 사용하여 자동으로 코드를 생성한 후, Code Inspector로 모델과 코드와의 일치 여부와 추적성(Traceability)을 확인하는 기능을 제공한다[3]. 코드 생성 시, 신호 및 파라미터와 Stateflow에 대한 코드 최적화 옵션을 설정할 수 있다. Inline Parameter, Signal Storage Reuse 등의 최적화 옵션을 설정하면, 코드 생성 과정에서 모델 블록과 코드가 정확히 일치하지 않을 가능성이 있다.[4] Simulink에서는 모델과 코드의 추적성 확인을 위해 Code Inspector가 제공되고 있으나 일부 블록 모델로 제한되어 있어, 모델 검증과는 별도로 개발자가 직접 코드에 대한 검증을 수행해야 한다.

Simulink의 Design Verifier와 모델 커버리지 분석(Model Coverage Analysis) 도구와는 별도로 코드의 신뢰성 시험을 위해 코딩규칙(Coding Rule) 및 실행시간 오류(Run Time Error)를 확인하는 정적시험(Static Testing)과 구조적 SW 시험 방법인 코드실행률(Code Coverage)을 확인하는 동적시험(Dynamic Testing)이 필요하다.

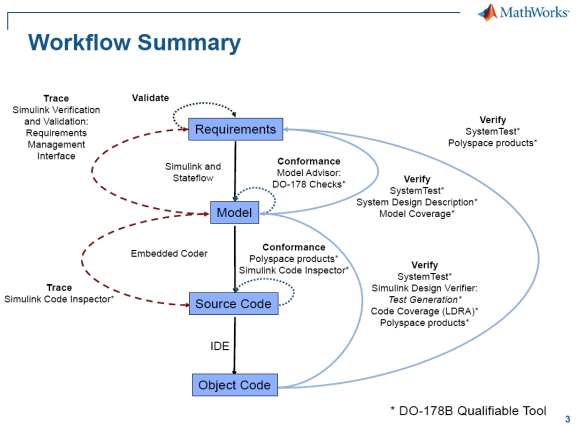


Fig. 1. SW development process based on Simulink models[5]

2.1.2 MAAB 스타일 가이드라인

MAAB 스타일 가이드라인은 1998년에 포드, 크라이

슬러, 도요타 등 자동차 제조 회사의 자사, 협력사 및 하청업체 사이에 모델 기반 설계 업무 협력 시, 프로젝트 성공을 목적으로 마련되었다. 버전(이하 V) 1.0 가이드라인 배포를 시작으로 현재 V3.0으로 개정되었다. MAAB의 주요 목표는 시스템 통합 시 개발 산출물(모델, 코드, 문서 등)을 통일화하여 읽기 쉽고 재사용이 가능한 무결성 모델을 생성하여 간편하고 효과적인 모델 개발 프로세스를 만드는 것이다[6].

MAAB는 Table 1에서 제공된 양식을 사용하여 SW 환경, 명명규칙, 모델 아키텍처, Simulink 모델, Stateflow 모델, 열거형 데이터(Enumerated Data), MATLAB 함수로 분류된 6개 항목으로 모델 스타일 가이드라인을 명시하고 있다. 특히 Rationale의 확인(Validation) 및 검증(Verification), 코드생성(Code Generation)과 관련된 가이드라인은 모델 설계 과정에서 코드의 결함을 미리 방지할 수 있는 가능성을 제공한다.

Table 1. The form of MAAB style guideline

ID: Title	ID: Title of the guideline
Priority	• madatory(필수) • strongly recommended(강한 권고) • recommended(권고)
Scope	• MAAB(공통) • NA-MAAB(북미) • J-MAAB(일본) • 특정 회사 등
MATLAB Version	가이드라인 적용 MATLAB 버전
Prerequisites	현재 가이드라인의 선행조건
Description	가이드라인 상세 설명
Correct/ Incorrect	현재 가이드라인 준수 시, 올바른 사용법과 잘못된 사용법의 예와 그에 따른 이익과 불이익 설명
Rationale	• Readability(R) :알고리즘의 빠른 이해 • Workflow(W) :효과적인 개발 프로세스와 작업 흐름 • Simulation(S) :효율적인 시뮬레이션 및 분석 • Verification & Validation(V&V) :모델과 코드의 검증 능력 • Code Generation(CG) :내장형 시스템에 적합한 코드 생성
Last Change	최종 가이드라인 개정문서 버전

2.2 무기체계 내장형 SW 신뢰성 시험

무기체계 내장형 SW 획득 및 관리 지침서(이하 지침서)는 대한민국 방위사업청(DAPA) 무기체계 연구개발 사업으로 개발되는 내장형 SW의 품질 및 신뢰성 향상을 위해 프로그램 코딩규칙 및 실행시간오류를 포함하는 정적시험과 SW 구조 기반의 동적시험 평가 지침을 다루고 있다[1]. 특히 동적시험 평가 기준은 항공기의 기능 및 성능 안정성과 관련된 국제 감항인증 기준인 DO-178B를 참조하여 문장, 결정, MC/DC 커버리지를 시험 판단 기준으로 정하고 있다.

2.2.1 정적시험

무기체계 내장형 SW의 정적시험 방법 중 코딩규칙의 적용범위는 C/C++로 작성된 무기체계 개발, 전장관리정보체계 개발 SW 등 방위사업으로 추진 및 개발되는 SW를 대상으로 한다. 주요 지침 내용은 주석 및 명명 규칙과 프로그램 코딩규칙이며, 코딩규칙은 Table 2와 같이 총 65개로 구성되어 있다. SW의 결함 정도별 판단 기준은 치명적인 고장 및 오작동[1] 발생 여부와 발생 빈도에 따라 치명(Critical), 경고(Major), 권고(Minor)로 분류한다.

Table 2. The coding rules in Wespon System Embedded SW

분류		주요내용	항목수
C, C++ 공통 적용	스타일	다중 return문 자제 등 스타일 관련 규칙	11
	초기화	변수, 배열, 포인터 등 초기화 관련 규칙	3
	식별자	전역변수 중복 자제 등 식별자 관련 규칙	4
	조건식	고정된 판단 조건문 등 조건식 관련 규칙	5
	변환	함수, 데이터 형 변환 등 변환 관련 규칙	8
	포인터/ 배열	NULL 포인터 참조, 초과된 배열 인덱스 사용 등 포인터 및 배열 관련 규칙	5
	연산자	0 나누기 연산 등 연산자 관련 규칙	9
C 전용		scanf, malloc 등 C언어 전용 코딩 규칙	5
C++ 전용		예외처리, 생성/소멸자 등 C++언어 전용 코딩 규칙	15

코딩규칙 확인 외에 다른 정적시험은 버퍼 overrun/underrun, Null 포인터 참조, 메모리 누수(Leak) 등 SW 실행 중 프로그램 동작에 영향을 미칠 수 있는 실행시간오류를 검출하고 방지하는 것을 목표로 한다. 실행시간오류의 시험평가 기준으로 미국 MITRE에서 작성한 CWE-658/659 오류 목록을 선별하여 사용할 수 있다. CWE-658은 C, CWE-659는 C++ 코드에 대한 실행시간오류를 다루고 있으며, 초기화 되지 않은 변수를 사용하거나, 할당되지 않은 포인터 참조 등 일부 코딩규칙 항목과 중복되는 오류가 존재한다. 실행시간오류 결함의 판단 기준은 중(Major)과 경(Minor) 결함 정도로 나누고, 거짓 경보(False Alarm)는 실행시간오류에 포함하지 않는다.

2.2.2 동적시험

지침서에서 요구하는 동적시험은 요구사항, 코드 구조, 사용자의 경험 기반 동적시험 방법 중 SW 구조 기반 분석으로 코드실행률(단위[%])을 시험 판단 기준으로 사용된다.

코드실행률의 목표값 설정 시 SW의 임무 중요도, 기능 안전성 및 통제능력, 사용빈도를 고려하여 설정한다. 코드실행률의 목표를 달성하기 위해서 순환복잡도(Cyclomatic Complexity)등 SW 구조의 복잡도를 고려해서 설계를 수행해야 한다. SW 구조 복잡도는 코드의 독립적인 수행 경로의 수로 결정되며, 반복문 및 분기 조건의 수가 증가할 수 록 복잡도가 높아져 코드실행률을 만족하는 테스트케이스(Test Case)의 생성이 어려워진다.

코드실행률은 시험 수행 강도와 방법에 따라 다양하다. 대표적으로 조건 분기문에서 조건의 참과 거짓에 따라 테스트케이스를 생성하고 시험을 수행하는 문장(Statement), 결정(Decision/Branch), 조건(Condition), MC/DC(Modified Condition/Decision Coverage) 등 다양한 커버리지가 있으며, MC/DC를 만족하는 테스트케이스는 조건 및 결정 커버리지를 포함한 하위 모든 커버리지를 만족하는 동적시험 방법이다[7]. 또한 c-use, p-use, all-use 등 데이터 흐름(Data Flow)에 따라 테스트케이스를 설계하는 데이터 흐름 기반 커버리지도 구조 기반 테스트 기법 중 하나지만, 지침서에서는 DO-178B 레벨 A[8]를 기반으로 문장, 결정, MC/DC 커버리지를 시험 판단 기준으로 정하고 있다.

2.3 MAAB 기반 모델 가이드라인 제안

본 절에서는 군용 항공기에 탑재되는 비행조종컴퓨터 OFP 개발 시, Simulink를 이용한 모델 기반 개발을 적용하기 위하여 MAAB 스타일 가이드라인 항목 중 지침서에서 제안하는 무기체계 내장형 SW의 신뢰성 시험 방법 및 기준과 연관된 항목을 확인한 결과를 Table로 정리하였다. 해당 항목들이 모델에서 자동으로 생성된 코드의 정적시험과 동적시험에 미치는 영향성을 분석하여 DAPA의 SW 신뢰성 시험을 만족하는 항목을 선정하는 기준을 제시하였다.

2.3.1 정적시험 만족을 위한 가이드라인

Table 3에서 MAAB 가이드라인 항목 중 지침서의 코딩규칙과 연관된 항목을 확인하여 그 결과를 정리한 것이다. Table 3의 MAAB 분류 항목 및 ID에 대응하는 코딩규칙은 '지침서 코딩규칙 항목'에서 확인할 수 있다.

Simulink의 Model Advisor를 사용하여 MAAB 가이드라인의 적합성 여부를 자동으로 확인할 수 있는지 그 여부를 'Model Advisor 자동 확인 여부' 항목에 나타내었다. Simulink에서는 총 110개의 MAAB 항목 중 49개가 Model Advisor에서 자동으로 확인 가능하다. C언어로 작성된 코드의 실행시간오류 검출을 위해 지침서에서는 CWE-658을 결함 판단 기준으로 제시하였다. CWE-658 V2.1 기준으로 총 80개의 항목 중 지침서의 실행시간오류 항목은 Weakness 항목인 74개를 대상으로 프로젝트의 특성에 따라 선별하여 사용할 수 있다[9].

MAAB 가이드라인은 지침서의 코딩규칙에 비해 실행시간오류와 직접 관련된 항목수가 적지만, 코딩규칙과 연관되어 발생할 수 있는 실행시간오류가 많기 때문에 MAAB를 준수하는 것이 실행시간 오류를 줄이는데 도움을 준다.

실행시간 오류와 MAAB 스타일 관련 항목을 살펴보면 실행시간 오류 항목 중 사용하지 않는 변수와 dead code 오류는 na_0036(Default variant - subsystem)와 db_0081(Uncorrected signals and block inputs/outputs)로 검증할 수 있다. 추가로 함수의 리턴값을 처리하지 않는 문제는 jc_0511(Setting the return value from a graphical function)로 인해 발생 가능하다. 또한 na_0003(Simple logical expressions in If Condition block), na_0036(Default variant) 등 Model Advisor에서 자동으로 확인할 수 없는 MAAB 항목은 가이드라인의 설명(Description)과 적합(Correct)/부적합(Incorrect) 예시를 참

조하여 MAAB 가이드라인의 부합 여부를 확인할 수 있다.

지침서의 코딩규칙은 ISO 26262, IEC 61508, MISRA C:2004 등 여러 코딩규칙 관련 표준과 연관되어 있다. Simulink Model Advisor에서도 MAAB 가이드라인 검증 방법과 동일하게 MISRA C:2004, ISO 26262 등 모델링 규칙을 검증할 수 있으나, 규격의 일부만을 검증하거나 검증이 가능한 모델 블록들이 제한적이다. 특히 MISRA C:2004 검증 항목은 'Check for blocks not recommended for MISRA-C:2004 compliance' 체크리스

Table 3. The relative coding rules of MAAB

분류	ID	Guideline Title	Priority	지침서 코딩규칙	Model Advisor 자동 검증 여부
명명규칙 (Naming Conventions)	ar_0001	Filenames	필수	명명규칙	○
	na_0035	Adoption of naming conventions	권고	명명규칙	×
	jc_0211	Usable characters for Input block and Output block	강한 권고	명명규칙	○
	jc_0221	Usable characters for signal line name	강한 권고	명명규칙	○
	na_0030	Usable characters for Simulink bus names	강한 권고	명명규칙	×
	jc_0231	Usable characters for block names	강한 권고	명명규칙	○
모델 아키텍처	na_0020	Number of inputs to variant subsystems	권고	C전용-4	×
	na_0036	Default variant	권고	스타일-9	×
Simulink	jc_0061	Display of block names	권고	명명규칙	○
	jc_0281	Naming of trigger port block and enable port block	강한 권고	C전용-4	○
	na_0009	Entry versus propagation of signal labels	강한 권고	식별자-2	○
	db_0081	Uncorrected signals and block inputs/outputs	필수	스타일-8 조건식-5	○
	na_0003	Simple logical expressions in If Condition block	필수	연산자-8	×
	na_0002	Appropriate implementation of fundamental logical and numerical operations	필수	조건식-4 연산자-8	×
	na_0011	Scope of goto and from blocks	강한 권고	스타일-7	○
Stateflow	db_0123	Stateflow port names	강한 권고	식별자-3	○
	db_0137	States in state machines	필수	스타일-9	○
	jc_0501	Format of entries in a state block	권고	스타일-10	○
	jc_0511	Setting the return value from a graphical function	필수	스타일-8	○
	na_0001	Bitwise Stateflow operators	강한 권고	연산자-7	○
	jc_0451	Use of unary minus on unsigned integers in Stateflow	권고	연산자-5	○
	jc_0481	Use of hard equality comparisons for floating point number in Stateflow	권고	조건식-1	○
	db_0151	State machine patterns for transition actions	강한 권고	스타일-10	○
	db_0149	Flowchart patterns for condition actions	강한 권고	스타일-10	×
열거형 데이터	na_0031	Definition of default enumerated value	권고	C전용-5	×
MATLAB 함수	na_0025	MATLAB function header	강한 권고	주석규칙	×
	na_0034	MATLAB function block input/output settings	강한 권고	스타일-2	×

트를 제공하고 있지만, 소스의 적합성 판단을 ‘Simulink Block Support Table’에서 code generation에 해당하는 블록으로 제한하고 있다. 또한 ‘Check configuration parameters for MISRA-C:2004 compliance’에서는 소스 코드 생성 관련 환경설정 문제를 다루고 있어, 실제 모델 설계 스타일 작성 규칙을 위한 MAAB 검증 항목과는 검증 방법이 다르다[10].

Table 3과 같이 각 MAAB을 따라 개발된 모델의 소스코드는 지침서의 코딩규칙과 CWE-658의 실행시간 오류를 미연에 방지하여 무기체계 내장형 SW의 신뢰성을 향상시킬 수 있다.

2.3.2 동적시험 만족을 위한 가이드라인

MAAB 기반 동적시험에서는 MAAB 스타일 가이드라인에서 조건 분기에 관련된 항목을 대상으로 모델과 코드에 대한 동적분석을 수행한다. 문장, 결정, MC/DC 커버리지는 조건 분기와 깊이 연관되어 있다. 특히 Simulink 모델에서 결정 커버리지는 Function Header, If, Switch, For, While 블록을, MC/DC 커버리지는 If, While 블록을 고려해야 한다[11].

Table 4에서 If, Switch 등 조건 분기와 직접 관련된 MAAB 항목을 나타내었다. Table 4에서 ‘Rationale’이 ‘Verification and Validation’ 또는 ‘Code Generation’과 관련된 MAAB 항목의 영향성을 분석하였다. 그리고 ‘코드 커버리지 관련항목’에서는 MAAB 가이드라인 항목이 코드의 조건 분기문에 포함된 로직 또는 코드 구조의 복잡도에 대한 영향성에 따라 ‘로직’과 ‘복잡도’로 기준을 정했다.

Table 4. The conditional branch guidelines in MAAB

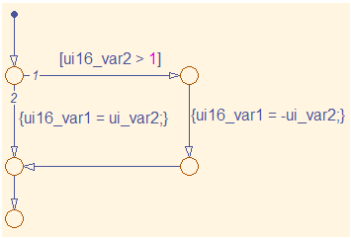
분류	ID	Guideline Title	Rationale	코드 커버리지 관련항목	Model Advisor 자동 검증 여부
모델 아키텍처	na_0037	Use of single variable variant conditionals	R/S/CG	로직	×
	na_0036	Default variant	R/S/CG	로직	×
Simulink	na_0002	Optimization parameters for Boolean data types	W/CG	로직	×
	jc_0141	Use of the switch block	R/W/V&V/CG	로직	○
	na_0012	Use of switch vs. If-Then-Else action subsystem	R/V&V/CG	복잡도	×
	db_0114	Simulink patterns for If-Then-Else-if constructs	R	영향성 없음	×
	db_0115	Simulink patterns for case constructs	R	영향성 없음	×
	na_0028	Use of If-Then-Else action subsystem to replace multiple switches	R	영향성 없음	×
	na_0003	Simple logical expression in If Condition block	R/W/CG	복잡도	×
Stateflow	na_0013	Comparison operation in Stateflow	S/V&V/CG	로직	○
	db_0150	State machine patterns for conditions	R	영향성 없음	×
	db_0134	Flowchart patterns for If constructs	R/W/S/V&V/CG	로직	×
	db_0135	Flowchart patterns for loop constructs	R	영향성 없음	×
	na_0038	Levels in Stateflow charts	R	영향성 없음	×
	na_0040	Number of states per container	R/V&V/CG	복잡도	×
	db_0137	States in state machines	R/W/V&V/CG	로직	○
	jc_0481	Use of hard equality comparisons for floating point number in Stateflow	V&V/CG	로직	○
MATLAB 함수	na_0018	Number of nested If/Else and case statement	R/CG	복잡도	×
	na_0022	Recommended patterns for switch/case statements	S/V&V/CG	로직	×

2.4 제안한 모델 가이드라인의 영향성 검증

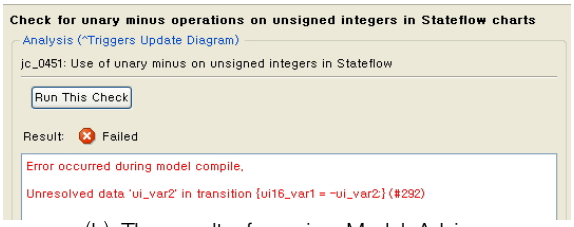
제안한 모델 가이드라인이 SW 신뢰성 향상에 미치는 영향성을 확인하기 위해 MATLAB/Simulink R2012b 개발 도구와 LDRA V9.1.0 시험 도구를 사용하였다. Simulink 모델에서 자동으로 C코드 생성을 위해 Realtime Workshop Embedded Coder를 사용하였다. 모델로부터 코드 생성 시 최적화 과정에서 발생하는 모델 커버리지와 코드 커버리지의 차이를 줄이기 위해, Inline, Re-use, Dead Code Optimization을 수행하지 않도록 설정하였다.

2.4.1 정적시험 영향성 분석

MAAB 가이드라인과 지침서의 코딩규칙 및 실행시간 오류의 연관성을 분석하기 위해 MAAB 가이드라인에 부적합한 모델을 Table 3의 ID별로 생성하여 Model Advisor와 LDRA 정적분석 도구(Static Analysis)를 사용하였다.



(a) An incorrect model



(b) The result of running Model Advisor

```
494    1    /* Chart: '<Root>/Chart' incorporates:
495    1     * Inport: '<Root>/ui16_var2'
496    1     */
497    1    /* Gateway: Chart */
498    1    /* During: Chart */
499    1    /* Entry Internal: Chart */
500    1    /* Transition: '<S1>:11' */
501    1      if
502    1        (
503    1          jc_0451_U.ui16_var2 > 1
(C) STATIC VIOLATION     : 434 S : Signed/unsigned conversion without cast. : (unsigned short and int): jc_0451_U.ui16_var2
504    1        )
-------------------------------------------------------------------------------------------------------------------
505    2        {
506    2    /* Transition: '<S1>:10' */
507    2    /* Transition: '<S1>:9' */
508    2          tmp_0 = - jc_0451_DWork.ui_var2 ;
509    2          if
510    2            (
511    2              tmp_0 < 65536.0
512    2              )
```

(c) The result of running LDRA Static Analysis

Fig. 2. The static analysis on jc_0451

Fig. 2는 jc_0451 항목에 대한 정적분석 결과로 (a)와 같은 jc_0451 가이드라인에 부적합한(Incorrect) 모델을 생성하여, (b) Model Advisor 검증 결과와 (c) LDRA 정적분석 결과를 나타낸 것이다. jc_0451은 지침서의 연산자-5에 연관된 항목으로 데이터형이 'unsigned'인 데이터에 '-'를 사용하여 결과를 할당하는 것을 금지함으로써 데이터에 의도하지 않은 변화를 막는 것이 목적이다. Fig. 2의 (b)에서 Model Advisor의 jc_0451 자동 검증 수행 시 'Fail'이 발생했으며, LDRA에서도 관련 변수인 ui16_var2에 대해 '434 S: Signed/unsigned conversion without cast. : (unsigned short and int) : jc_0451_U.ui16_var2' 오류가 발생하였다.

2.4.2 동적시험 영향성 분석

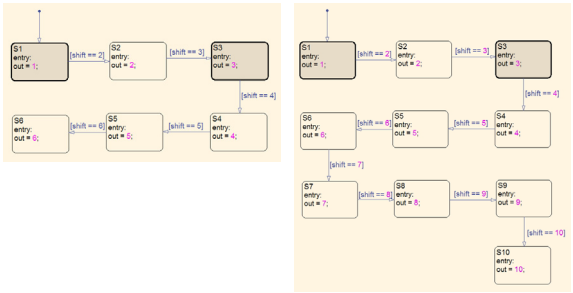
검증하고자 하는 모델에 대해 모델 커버리지 분석을 수행하고, Embedded Coder를 활용하여 모델로부터 생성된 소스코드에 대해 LDRA에서 동적분석을 수행한 결과를 비교한 것이다. LDRA에서는 조건 분기 로직과 복잡도 분석을 위해 Complexity Analysis, 테스트케이스 분석을 위해 MC/DC Test Case Planner 및 Dynamic Coverage Analysis를 사용하였다.

2.4.2.1 복잡도

na_0040 항목은 상태기계를 활용한 Stateflow Chart 모델 설계 시, 한 개의 컨테이너(Container; State)에 포함된 상태의 적정 상태수를 6개에서 10로 제한하고 있다. na_0040 항목이 모델 커버리지 분석과 코드 커버리지에 미치는 영향성을 확인하기 위해 Fig. 3과 같은 순차적 상태 Stateflow Chart를 생성한 후 비교 분석을 수행하였다.

Fig. 3의 (a) 6단계 순차적 상태에서 (b) 11단계 순차적 상태로 1단계씩 순차적 상태수를 증가시켜 모델을 생성한 후, 모델 커버리지 분석과 LDRA 동적분석 결과를 Table 5에 나타냈다. Table 5의 결과에서 알 수 있듯이, 상태수가 6에서 11으로 증가함에 따라 순환복잡도가 2씩 증가했으며, 모델에서 결정 커버리지에 필요한 테스트케이스수인 결정 결과수와 코드의 결정포인트수가 증가하였다.

결정 결과수는 Fig. 4 (a)의 Decision Outcomes를, 결정 포인트수는 (c)에서 LDRA Dynamic Coverage Report에서 'Branch/Decision Coverage Profile'의 Code Processing Decision Point를 참조하였다. 상태수 증가뿐만 아니라 조건식에 포함된 개별 조건수의 증가도 복



(a) The 6-step sequential states (b) The 11-step sequential states

Fig. 3. The examples for sequential states on na_0040

Summary

Model Hierarchy/Complexity:	Test 1 D1	
1. model_ldra_11state	22	100%
2. . . . State11	21	100%
3. . . . . SF: State11	20	100%

Details:

1. Model "model_ldra_11state"

Child Systems: State11

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	1	22
Decision (D1)	NA	100% (31/31) decision outcomes

(a) The result of running Model Coverage Analysis

Complexity Metrics (model_ldra_11state.c)

Procedure	Knots	Cyclomatic Complexity
model_ldra_11state_step	56 (F)	22 (F)
model_ldra_11state_initialize	0 (P)	1 (P)
model_ldra_11state_terminate	0 (P)	1 (P)
Total for model_ldra_11state.c	56 (F)	22 (P)

(b) The result of running LDRA Complexity Analysis

LINE NUMBERS: FROM	REFORMATTED (SOURCE) TO	PREVIOUS RUNS	CURRENT RUN	COMBINED	CODE PRECEDING DECISION POINT
511 (57)	512 (57)	1	1	2	.is_active_c4_model_ldra_11state == 0)
511 (57)	523 (67)	50	50	100	
522 (67)	672 (194)	1	1	2	else
527 (68)	528 (69)	50	50	100	_DWork.is_c4_model_ldra_11state) {
527 (68)	542 (81)	0 ***	0 ***	0 ***	
527 (68)	558 (93)	0 ***	0 ***	0 ***	
527 (68)	572 (108)	0 ***	0 ***	0 ***	
527 (68)	586 (120)	0 ***	0 ***	0 ***	
527 (68)	600 (132)	0 ***	0 ***	0 ***	
527 (68)	614 (144)	0 ***	0 ***	0 ***	
527 (68)	628 (156)	0 ***	0 ***	0 ***	
527 (68)	642 (168)	0 ***	0 ***	0 ***	
527 (68)	656 (180)	0 ***	0 ***	0 ***	
533 (71)	534 (71)	0 ***	0 ***	0 ***	model_ldra_11state_U.in == 2.0)
533 (71)	541 (79)	50	50	100	
541 (79)	671 (192)	50	50	100	break ;
547 (83)	548 (83)	0 ***	0 ***	0 ***	model_ldra_11state_U.in == 11.0)
547 (83)	555 (91)	0 ***	0 ***	0 ***	
555 (91)	671 (192)	0 ***	0 ***	0 ***	break ;
557 (94)	671 (192)	0 ***	0 ***	0 ***	break ;
563 (98)	564 (98)	0 ***	0 ***	0 ***	model_ldra_11state_U.in == 3.0)
563 (98)	571 (106)	0 ***	0 ***	0 ***	
571 (106)	671 (192)	0 ***	0 ***	0 ***	break ;
577 (110)	578 (110)	0 ***	0 ***	0 ***	model_ldra_11state_U.in == 4.0)
577 (110)	585 (118)	0 ***	0 ***	0 ***	
585 (118)	671 (192)	0 ***	0 ***	0 ***	break ;

(c) The result of running LDRA Dynamic Coverage Analysis

Fig. 4. The dynamic analysis of the 11-step sequential states

Table 5. The dynamic analysis of the sequential state models

모델 유형	모델 커버리지 분석		LDRA 동적분석	
	복잡도	결정 결과수	복잡도	결정 포인트수
6 단계	12	16	12	14
7 단계	14	19	14	16
8 단계	16	22	16	18
9 단계	18	25	18	20
10단계	20	28	20	22
11단계	22	31	22	24

※ 복잡도: 순환복잡도(Cyclomatic Complexity)
결정 결과수: Decision Outcomes[11]
결정 포인트수: Code Processing Decision Point

잡도와 테스트케이스수를 증가[7]시키기 때문에 개별 조건수가 늘어날 경우 상태수를 더욱 줄여 설계하는 것이 복잡도와 테스트케이스의 수를 줄일 수 있다.

2.4.2.2 로직

db_0137 항목은 상태기계의 상태 스타일에 관한 규칙으로 상태 로직에 관련된 항목들을 포함하고 있다. 이 중에서 세 번째 항목은 계층적 상태 모델 설계 시, 초기 상태를 항상 조건없는 Default Transaction으로 제한하고 있다. 해당 가이드라인이 모델 커버리지와 코드 커버리지에 미치는 영향성을 확인하기 위해 Fig. 5와 같은 db_0137 부적합 모델을 생성하였다.

S2 상태의 Default Transaction이 shift(unsigned int형) 변수가 0보다 작은 '항상 거짓' 조건으로 설정하여 S3 상태가 수행될 수 없도록 구성하였다.

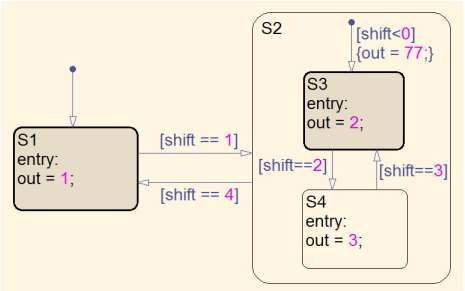
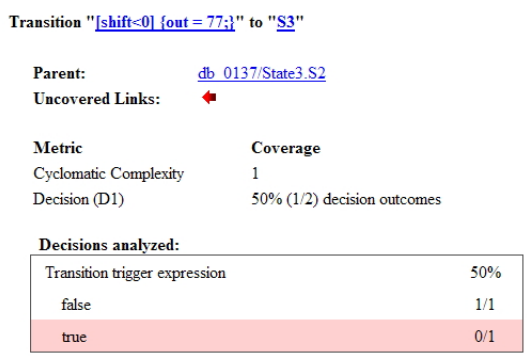
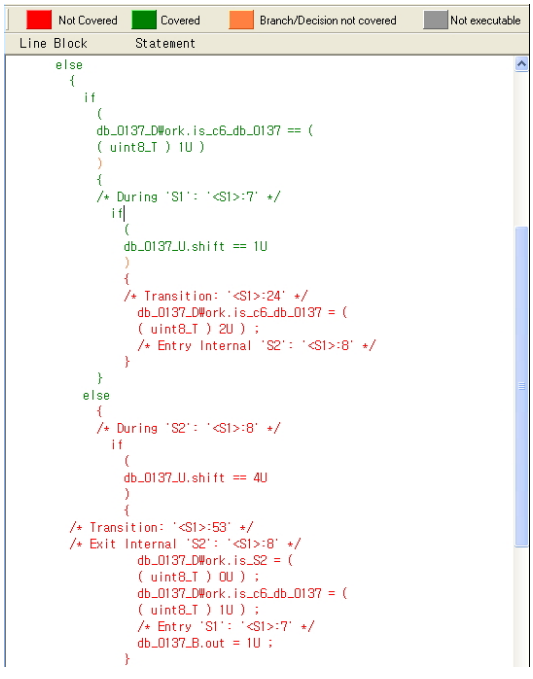


Fig. 5. An incorrect model on db_0137

Fig. 6는 db_0137의 부접합 모델의 모델 커버리지 분석과 LDRA에서 코드 레벨의 동적 커버리지 분석결과를 나타낸 것이다. 모델 커버리지 분석결과 참(True)인 조건을 수행할 수 없기 때문에 커버리지가 50%로 나타나고 있다. 모델에서 생성된 소스코드에 대해 LDRA의 커버리지 분석 결과 'Not Covered' 관련 코드에서 S2 상태를 수행할 수 없음을 확인하였다.



(a) The result of running Model Coverage Analysis

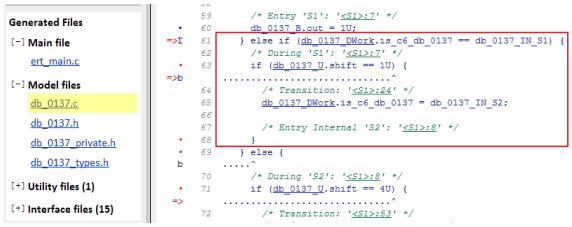


(b) The result of running LDRA Dynamic Coverage Analysis

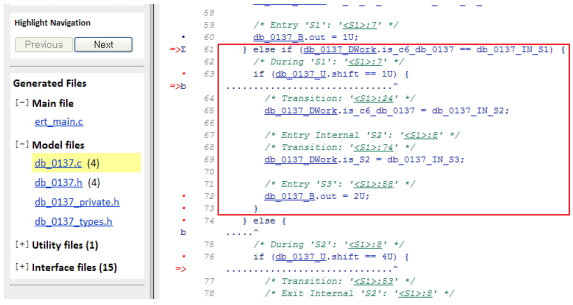
Fig. 6. The dynamic analysis of the incorrect model on db_0137

Fig. 7에서 db_0137 규칙 관련 적합 모델과 부적합 모델의 C 코드를 나타내었다. db_0137 적합 모델은 S2의 default transaction 조건을 삭제한 것이다. Fig 7에서 두 코드를 비교해 보면, (a)에서 존재하지 않았던 S2 상태 도입 코드가 (b)에서 'db_0137_DWork.is_S2 = db_0137_IN_s3;' 도입부가 추가 생성된 것을 확인할 수 있다.

상태 모델을 설계할 때 Default Transaction을 항상 거짓인 부적합 모델로 생성하여, 실제 결정 커버리지를 100% 달성 할 수 없는 소스코드가 생성되었다. 이번 검증 시험을 통해 Default Transaction의 조건 로직 오류가 문장, 결정 및 MC/DC 커버리지 100% 달성할 수 없는 원인 이 될 수 있음을 확인하였다. 모델 설계 시 Default Transaction 관련 분기 조건의 로직의 오류가 없는지 확인이 반드시 필요하다.



(a) The incorrect model



(b) The correct model

Fig. 7. The source code from the correct/incorrect model

3. 결 론

각 무기체계 분야마다 모델 기반 내장형 SW의 신뢰성 확보에 필요한 설계 가이드라인 항목 선정을 위한 연구가 필요하지만 국내 연구는 그 필요성에 비해

아직 연구 결과들을 찾기 어려운 실정이다.

본 논문에서는 비행조종컴퓨터 OFP의 모델 기반 개발을 위하혀 기존에 개발된 MAAB 스타일 가이드라인 항목을 이용하여 DAPA의 SW 신뢰성 요구조건을 만족할 수 있는 40개의 가이드라인 항목을 선정하였다. 각 가이드라인 항목에 대한 영향성 및 기준에 대한 검증은 Simulink R2012b와 LDRA V9.1.0 환경에서 DAPA의 신뢰성 시험과 관련된 MAAB 항목의 부적합 모델을 생성한 후, 정적시험 연관 항목은 Model Advisor와 LDRA 정적분석 도구로 그 결과를 확인하였으며, 동적시험 연관 항목은 LDRA 동적분석 도구로 조건 분기 로직과 복잡도를 기준으로 영향성을 검증하였다. Table 6과 같이 본 논문에서 제안한 40개의 가이드라인 중 27개 항목이 무기체계 내장형 SW 코딩규칙과 CWE-658의 일부 실행시간오류와 연관되어 있으며, 13개의 항목이 동적시험을 위한 코드 복잡도 및 논리 구조와 관련되어 있다.

Table 6. The number of MAAB relative to source code reliability testing

시험 분류	MAAB Priority	신뢰성 시험 관련 MAAB 항목수	
정적시험	필수	6	27
	강한 권고	13	
	권고	8	
동적시험	필수	4	13
	강한 권고	4	
	권고	5	

References

[1] DAPA, “무기체계 내장형 소프트웨어 획득 및 관리 실무 지침서,” 방위사업청(DAPA), 2011.

[2] Stephen A. Jacklin, “Certification of Safety-Critical Software Under DO-178C and DO-278A,” AIAA Infortech at Aerospace, 2012.

[3] The Mathworks, “Simulink Code Inspector User's Guide R2012b,” The Mathworks Inc., 2012.

[4] William Aldrich, “Using Model Coverage Analysis to Improve the Controls Development Process,” AIAA Modeling and Simulation Technologies Conference and Exhibit, 2002.

[5] http://www.mathworks.co.uk/company/events/conferences/matlab-tour/proceedings/model-based-design-for-do-178.pdf

[6] MAAB, “Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow Version 3.0,” MathWorks Automotive Advisory Board (MAAB).

[7] Aditya P. Mathur, Foundations of Software Testing, Pearson Education, pp. 402-491, 2008.

[8] RTCA Inc., “Software Considerations Airborne Systems and Equipment Certification,” Document RTCA/DO-178B, 1992.

[9] Steven M. Christey, Janis E. Kenderdine, John M. Mazella and et al., “Common Weakness Enumeration - A Community-Developed Dictionary of Software Weakness Types,” The MITRE Corporation, 2011.

[10] The Mathworks, “Modeling Guidelines for High-Integrity Systems R2012b,” The Mathworks Inc., 2012.

[11] The Mathworks, “Simulink Verification and Validation User's Guide R2012b,” The Mathworks Inc., 2012.