정형성 기반 국방 안전/보안필수 소프트웨어 개발 및 인증 기준

- 안전/보안필수 소프트웨어 인증 프로세스에 대한 정형기법 적용 방안 연구 -

Formalism-Based Defense Safety/Security-Critical Software Development & Certification Criteria

- Application of Formal Methods to Safety/Security-Critical Software Certification Process Activities -

김 창 진* 최 진 영*
Kim, Chang-Jin Choi, Jin-Young

ABSTRACT

The paper provides the approach to apply formal methods to the development and certification criteria of defense safety/security-critical software. RTCA/DO-178B is recognized as a de facto international standard for airworthiness certification but lack of concrete activities and vagueness of verification/certification criteria have been criticized. In the case of MoD Def Stan 00-55, the guidelines based on formal methods are concrete enough and structured for the defense safety-related software. Also Common Criteria Evaluation Assurance Level includes the strict requirements of formal methods for the certification of high-level security software. By analyzing the problems of DO-178B and comparing it with MoD Def Stan 00-55 and Common Criteria, we identity the important issues in safety and security space. And considering the identified issues, we carry out merging of DO-178B and CC EAL7 on the basis of formal methods. Also the actual case studies for formal methods applications are shown with respect to the verification and reuse of software components.

주요기술용어(주제어) : Formal Methods(정형기법), Safety-Critical(안전필수), Security-Critical(보안필수), Certification Criteria(인증기준), RTCA/DO-178B, MoD DS 00-55, Common Criteria

1. 머리말

미국과 유럽 선진국들의 경우 RTCA/DO-178B[1]나 Common Criteria(CC)[2]를 바탕으로 안전필수 또는 보안필수 소프트웨어의 개발과 인증에 대한 경험이 상당 수준 축적되었을 뿐만 아니라 그 동안 식별된 문제점을 인식하고 이미 새로운 개선 방향에 대한 연구를 진행하고 있다. 그에 반해 국내에서는 기존의 표준 자체에 대한 기본적인 이해와 연구의 부족은 물론 이러한 표준을 강제하거나 인증의 기준으로 제시

† 2006년 11월 6일 접수~2007년 1월 26일 게재승인
* 고려대학교(Korea University)
주저자 이메일 : cjkim@formal.korea.ac.kr

할 수 있는 수단조차 없다는 사실은 문제가 아닐 수 없다. 최근 국내에서도 Smart Card를 비롯한 보안필수 시스템 인증을 위해 CC의 도입을 추진하는 움직임이 있으나 EAL(Evaluation Assurance Level)5 이상의 고수준의 보안성 인증은 아직도 초기 연구 수준을 벗어나지 못하고 있다. 영국 국방성(MoD)의 경우 안전등급 시스템의 개발 및 획득을 위해 정형기법에 기반을 둔 UK Def Stan 00-55(DS 00-55)[3]를 적용해 왔다. 그리고 이 규정의 폐기 이후에도 차기 국방 표준인 DS 00-56을 적용함과 동시에 안전관련 소프트웨어에 대해서는 DO178B의 적용 및 그 적합성에 대한 연구를 병행하고 있다. 이제 국내에서도 실질적인 개발 가이드라인과 체계적인 인증 절차의 수립, 그리고 소프트웨어의 안전성과 보안성을 보장할 수 있는 개발환경의 조성이 시급하다. 특히 안전성 및 보안성 요구가 높은 국방 분야 소프트웨어 개발의 오류를 최소화함과 동시에 명확한 인증 프로세스를 지원할 수 있는 가이드라인의 필요성은 절대적이라 하겠다.

안전성과 보안성의 인증은 실질적으로 서로 다른 인증체계를 통해 이루어지고 있다. 그러나 각각의 인증 기준을 살펴보면 기술적으로 상호 공유할 수 있는 부분들이 상당수 존재한다. 또한 안전필수 소프트웨어와 보안필수 소프트웨어의 기능적 성격은 분명히 차이가 있으나 이들이 요구하는 소프트웨어의 정확성과 완전성의 수준은 크게 다르지 않다. 이에 본 논문에서는 안전성 인증표준인 DO-178B, DS 00-55와 함께 보안성 인증표준인 CC에 대한 연구를 통해 취약점을 상호보완하고 핵심요소들을 병합함으로써 안전성과 보안성 인증 기준을 동시에 충족할 수 있는 통합 가이드라인을 제시한다.

2. 시스템 안전성에 대한 인식과 추세

군용 시스템의 경우 안전성과 보안성을 별개의 속성으로 분리하기 힘든 경우가 대부분인데, 이들 속성을 확보하기 위한 개발 과정상의 요구사항 또한 유사한 성격의 항목들이 많다. 따라서 본 논문에서는 먼저 DO-178B를 중심으로 안전성 위주의 문제 분석을 실시하고 그 결과를 토대로 CC와의 기준 병합을 통해 보안성과 안전성 요건을 동시에 만족하는 기준을 제시하고자 한다.

시스템의 안전성과 관련된 산업계의 변화는 주로 안전성 관련 규제정책, 시스템의 안전성 관련 위험요소에 대한 사회적 인식, 증가된 제품의 기능성 요구, 그리고 시스템 또는 소프트웨어 공학적 기술 발달과 같은 요인들이 주도해왔다.

대표적인 안전필수 시스템인 항공관제 시스템이나 철도 신호 시스템의 경우 그 안전성에 대하여 원칙기반(Rule-Base) 규제로부터 위험기반(Risk-Based) 규제로 정책 기조를 변화하는 추세이다[4]. 일반적으로 위험요소 기반의 규제정책들은 해당 시스템에 부여된 안전등급 목표에 맞추어 공급자들이 시스템 또는 서비스에 대하여 위험관리를 책임짐은 물론 위험관리를 적절히 수행했음을 입증하기 위한 데모를 요구하고 있다.

이와 더불어 시스템 및 소프트웨어의 요구사항은 다양한 측면에서 복잡해지고 있는데 그 대표적 요인 중 하나는 기하급수적으로 증가하는 시스템의 복잡도에 있다. 현재 운용중인 군용 전투기에는 대략 3~5백만 라인의 소프트웨어 코드들이 탑재되어 있는데 차세대 전투기의 경우 1000만 라인 이상의 소프트웨어 코드가 탑재될 것이다[4].

그와 동시에 시스템 내부의 통합화 추세는 하부 시스템간의 인터페이스를 더욱 복잡하게 만들고 있다. 자동차 크루즈 시스템의 예를 들면 크루즈 기능이 정상적으로 작동하기 위해서는 엔진뿐 아니라 트랜스미션 및 브레이크 시스템과도 인터페이스할 수 있어야 한다. 따라서 특정 기능을 분석하기 위해서는 연결된 모든 시스템에 대한 지식이 필요하며 다른 시스템의 변경에 의해 원래의 기능이 영향을 받기도 한다.

이와 같이 복잡하고 고도로 통합된 시스템일수록 그 정상적 동작 수행 과정에서 사용자의 역할이 차지하는 비중은 점차 줄어든다. 즉, 사용자의 운영상의 테크닉에 대한 의존도가 낮아질 뿐만 아니라 사용자가 시스템의 안전한 동작에 개입할 수 있는 여지도 줄어들었다. 반대로 시스템 자체에 대한 의존도와 시스템의 자율성(Autonomy)은 급격히 증가하고 있다. 스텔스 기능으로 유명한 F-117 전투기나 유로파이터 전투기의 경우 첨단 전투 성능 이면에 항공 역학적으

로는 매우 불안정한 기체를 보유하고 있기 때문에 더 이상 조종사의 감각이나 스킬만으로는 비행 자체가 불가능하다. 이들 전투기의 비행은 전적으로 컴퓨터 시스템의 통제 하에 이루어진다고 보아야 할 것이다. 위와 같은 변화의 추세는 시스템을 보다 효과적으로 개발하는 방법, 그리고 완벽한 안전성을 보장함과 동시에 안전성에 대한 분석이 가능하도록 설계하는 방법을 종용하고 있다.

3. RTCA/DO-178B

3장에서는 대표적인 소프트웨어 안전성 인증 기준인 RTCA/DO-178B의 주요 항목들을 살펴봄으로써 현재 어떤 사항들이 소프트웨어의 안전성 보장을 위해 요구되고 있는지 알아보고 그 문제점을 분석한다.

DO-178B는 통신, 항법, 감시 및 항공관제 시스템 관련 표준을 연구해온 미국의 사설 기관인 RTCA (Radio Technical Commission for Aeronautics)사가 발간한 항공기 및 장비 탑재 소프트웨어 시스템의 개발 가이드라인으로서 항공분야 안전성 인증에 있어 사실상(de facto) 의 국제표준으로 자리 잡고 있다.

가. DO-178B의 목적 및 범위

DO-178B는 항공기 관련 부품 및 탑재시스템의 소프트웨어 생산 가이드라인으로서 감항성(Airworthiness) 즉, 항공기 운항의 안전성과 관련된 요구사항 및 그 충족 기준들을 제시하고 있다.

DO-178B는 소프트웨어의 감항성 인증 측면을 기술하기 위해 시스템 수명주기와 소프트웨어 수명주기의 연관관계, 소프트웨어 인증 프로세스 이해를 위한 설명들을 포함하고 있다. 그러나 시스템 수명주기 프로세스 자체에 대한 설명은 포함시키지 않았으며 소프트웨어 범위를 벗어난 시스템의 안전성 평가나 검증 프로세스, 엔진 인허가 등도 다루지 않는다. 또한 생산된 소프트웨어의 운영적 측면을 배제하였으며 사용자가 변경할 수 있는 데이터에 대한 인증이나 인적자원, 조직 구조, 피 인증조직과 부품제조업자의 관계, 책임 분할 등 인적자원의 품질 관련 사항은 포함하지 않는다.

문서 구조상으로 볼 때 DO-178B는 소프트웨어 개발과 관련된 시스템적 측면(Section 2), 항공기 및 엔진 인증의 개요(Section 10) 그리고 소프트웨어 수명주기 프로세스(Section 3~12)에 대한 내용으로 구성되며 수명주기 데이터 및 기타 고려사항들을 별도로 다루고 있다. DO-178B의 수명주기 중에는 계획 및 개발프로세스와 별도로 검증, 형상관리, 품질보증, 인증 교섭 활동 명시하고 있는데 이들을 묶어 "필수 프로세스(Integral Process)"로 분류한다.

나. 소프트웨어 안전 등급과 Control Category

DO-178B는 시스템 안전평가 프로세스의 평가 결과를 통해 컴포넌트의 적정 등급을 구분한다. 주의할 사항은 소프트웨어 이외의 항목은 가이드라인의 범위를 벗어나지만 그 소프트웨어의 등급을 결정하는 것은 시스템 수준의 안전평가 프로세스라는 점이다.

DO-178B는 시스템의 실패 조건을 다음의 5가지로 나누고 있는데 소프트웨어의 실패가 시스템의 어떤 실패 조건을 유발하는가에 따라 소프트웨어의 등급이 결정된다.

- Catastrophic : 지속적 안전 비행 또는 착륙 불가
- Hazardous/Severe-Major : 시스템 안전성 또는 기능성의 격감, 승무원의 정상적 임무 수행 불가
- Major : 항공기의 성능 또는 승무원 임무 수행능력 저하로 인한 항공기 안전성 감소
- Minor : 심각한 안전성 저해 요건은 아니지만 안전성의 약간 감소 또는 불편 초래
- No Effect : 항공기 성능, 승무원 업무 부하에 영향을 주지 않음

[표 1] 시스템 실패조건과 소프트웨어 등급

Failure Condition	Software Level
Catastrophic	Level A
Hazardous/ Severe-Major	Level B
Major	Level C
Minor	Level D
No Effect	Level E

표 1은 시스템의 실패 조건과 그와 연관된 소프트웨어의 등급 간 상관관계를 나타낸 것이다. 예를 들어 특정 소프트웨어의 실패가 "Catastrophic" 실패 조건을 유발할 수 있다면 그 소프트웨어의 등급은 "Level A"에 해당한다.

소프트웨어 등급과는 별도로 DO-178B가 소프트웨어를 구분하는 또 하나의 기준은 소프트웨어 형상관리 통제 범위인데 형상관리 프로세스가 소프트웨어 수명주기 데이터를 어떤 목표를 가지고 관리 하는가에 따라 Control Category 1과 Control Category 2의 소프트웨어로 나눌 수 있다. 다음의 표 2는 DO-178B의 Control Category 1에 해당하는 형상관리 목표와 Control Category 2에 해당하는 형상관리 목표들을 구분해서 보여주고 있다. 즉, CC1 혹은 CC2 소프트웨어는 ● 표시된 목표에 따라 수명주기 데이터들을 형상관리 해야 한다는 것이다. 그리고 DO-178B의 감항인증을 필요로 하는 모든 소프트웨어들은 최소한 CC2 기준을 만족하는 형상통제를 수행해야 한다.

[표 2] SCM Process Objectives Associated with CC1 and CC2

SCM Process Objective	CC1	CC2
Configuration Identification	●	●
Baselines	●	
Traceability	●	●
Problem Reporting	●	
Change Control-integrity & Identification	●	●
Change Control - tracking	●	
Change Review	●	
Configuration Status Accounting	●	
Retrieval	●	●
Protection against Unauthorized Changes	●	●
Media Selection, Refreshing, Duplication	●	
Release	●	
Data Retention	●	●

다. DO-178B의 검증 프로세스

DO-178B의 정의에 따르면 검증 프로세스는 소프트웨어 개발 과정의 에러를 탐지하고 보고하는 과정이다. 그러나 에러의 제거는 검증이 아니라 개발 프로세스에 해당하는 활동이다[2].

다음의 표 3은 각 수준별 검증대상과 그 검증활동이 목표로 하는 충족 범위를 보여준다. 예를 들어 소프트웨어 High-Level 요구사항에 대한 검증은 시스템 요구사항에 대한 충족 여부를 확인할 수 있어야 하며 소프트웨어 아키텍처 또는 Low-Level 요구사항에 대한 검증을 통해 소프트웨어 High-Level 요구사항에 대한충족여부를 확인할 수 있어야 한다는 것이다. DO-178B에 명시된 검증 활동은 리뷰, 분석 및 테스트케이스의 생성과 테스트 절차 수행 등으로 구성되는데 소프트웨어 요구사항에 대한 모든 구현물과 각 검증 활동 간에는 추적성이 보장되어야 한다.

DO-178B에 명시된 대로 리뷰나 분석 활동을 수행하기 위해서는 피인증자(조직)가 소프트웨어의 요구사항과 아키텍처, 그리고 소스코드에 대한 정확성(Accuracy), 완결성(Completeness) 및 검증가능성(Verifiability)을 모두 입증해야 한다. 그러나 테스트를 통해 이를 입증하기 위해서는 테스트 케이스의 생성 과정에서의 내부적 일치성과 요구사항의 완결성에 대한 심층적인 분석이 필요하며 테스트 절차를 수행함에 있어 테스트 범위에 포함된 모든 요구사항의 충족 여부 또한 시연할 수 있어야 한다.

그러나 표 4에 나타난 바와 같이 요구사항 기반의 테스트를 통해 탐지해야 하는 에러의 종류는 다양한 반면에 일반적인 테스팅 과정에서 이와 같은 에러 전부를 완벽하게 찾아낸다는 것은 거의 불가능하다.

[표 3] 소프트웨어 검증의 수준별 검증 대상 및 충족 목표

Verification Target	Satisfaction Criteria
SW H-L Requirement	System Requirement
SW Archit./L-L Req.	SW H-L Requirement
Source Code	SW Archit./L-L Req.
Executable Object Code	SW Requirements

[표 4] 요구사항 기반 테스팅 방법론의 에러 탐지 범위

Methods	Objects	Error To Detect
HW/ SW 통합 테스팅	High Level Requirements	인터럽트 핸들링, 실행시간 요건, HW 실패에 대한 SW 반응, Data Bus 및 자원 연결 문제, 자체 고장 탐지 실패, 피드백 루프 오류, 메모리관리 HW 비정상 제어, Stack overflow, 필드적재 SW 보증 메커니즘 오류 등
SW 통합 테스팅	SW Requirements/ Architecture	변수/상수 비정상 초기화, 매개변수 전달 오류, 데이터 손상(전역데이터), 이벤트 및 동작 순서 오류
Low Level 테스팅	Low Level Requirements	알고리즘 오류, 비정상 루프, 로직 오류, 비정상적인 입력의 조합, 누락/손상 입력에 대한 비정상적 반응, 예외처리 실패, 계산 순서 오류, 알고리즘 정확도/성능 정밀도 오류

라. DO-178B 관련 이슈 분석

DO-178B가 항공분야 소프트웨어의 안전성 인증에 있어 사실상의 국제표준으로 자리 잡았지만 소프트웨어 수명주기 각 단계별 안전성 분석을 위해 제공되는 구체적 가이드라인이 부족하다는 점은 문제점으로 제기되어 왔다[5]. 인증 기준 충족을 위한 목표 중심적 표준의 성격이 강하기 때문에 어떻게 그 목표를 충족할 것인가는 전적으로 피인증자(조직)가 해결해야 할 과제인 것이다. 또한 테스팅이나 코드리뷰와 같은 분석 방법으로 소프트웨어 안전성과 관련된 모든 항목들을 완벽하게 분석하거나 그 안전성을 증명하기 힘들 뿐만 아니라 막대한 비용이 필요하다.

또 한 가지 산업계에서 가장 실질적인 이슈는 인증된 소프트웨어의 재사용과 그 인증 크레디트를 어떻게 재사용할 수 있는가 하는 것이다[6]. 일반적인 테스팅 방법론으로는 대상 소프트웨어 내에 에러가 없다는 사실을 증명할 수 없고, 한 번 인증 받은 소프트웨어가 다른 시스템의 일부로 재사용되거나 이미 제품화된 소프트웨어가 약간이라도 변경된 채로 시스템 내부에 다시 적재되었을 때 시스템 전체를 다시 인증받아야 할지 혹은 바뀐 소프트웨어만 재검증하면 되는지 그 검증 범위에 대한 명확한 해답을 제시하기가 어렵다.

이와 같은 비용, 일정의 부담, 그리고 그에 상응하는 완전성을 확보해야 하는 심각한 요구에 직면한 소프트웨어 개발 업체들은 다방면에서 독자적인 해결책을 제시하고 있다. 그러나 FAA/DER과 같은 인증기구 입장에서 시스템 개발 비용이나 일정 등은 안전성 인증에 있어 그다지 중요한 요소가 아니다[6]. 안전성의 보장만이 인증의 가부를 결정하는 유일한 기준인 것이다.

DO-178B 인증을 목표로 개발된 제품들 중 LynxOS-178은 철저한 컴포넌트의 독립을 통한 재사용의 예를 잘 보여준다. 즉, 안전성이 요구되는 시스템 상에서 어플리케이션과 디바이스 드라이버를 별도의 링크 과정 없이도 실행시킬 수 있도록 각각을 독립적인 모듈 단위 구성요소로 설치한 것이다. 따라서 실제로 애플리케이션과 디바이스는 상호간 의존성이 거의 없으며 디바이스 드라이버를 수정 할 필요가 있을 때에도 전체 소프트웨어 패키지를 재인증 받을 필요가 없다. 이러한 특징을 지니지 않는 소프트웨어는 모든 어플리케이션, 커널, 라이브러리를 포함한 전체 시스템에 대하여 단순 시스템 업그레이드, 보완에 대해서도 막대한 비용을 들여 재검증, 재인증을 받아야 한다.

4. DO-178B와 DS 00-55 비교 분석

DO-178B를 다른 표준들과 일대일로 대응시키거나 상호 비교하는 작업에는 제약이 따른다. DO-178B가 인증 프로세스를 지원하기 위한 소프트웨어 측면의 가이드라인을 표방하고 있고 DO-178B 문서상에 언급된 모든 항목들이 문서의 목표 범위 내에 포함되는 것은 아니기 때문에 DO-178B 단독으로는 전체 시스템 공정을 다루는 다른 표준들과 문장 혹은 패러그래

프 수준에서 일대일 매칭 시키기 어렵다. 그러나 DS 00-55의 경우에는 문서 범위가 국방 분야 안전관련 소프트웨어라는 점에서 DO-178B와 타깃의 유사성을 보이고 있으며 정형기법의 적용과 밀접하게 관련된 본 연구의 목적상 두 표준의 성격이 상호 보완적인 것으로 판단되어 검토 대상에 포함시킨다.

가. DO-178B와 DS 00-55의 차이점

DO-178B와 DS 00-55의 가장 큰 차이는 DO-178B가 안전 관리 혹은 위험 평가에 대한 언급을 하지 않는다는 것이다. 그러므로 DO-178B에는 'Safety Case'나 안전성 요건, 안전성 감사, 안전 관련 기록 유지 등 전형적인 안전 관리 시스템에서 다루어야 할 활동들이 빠져있다. 이는 DO-178B가 시스템 안전성 평가를 시스템 수명주기 활동으로 정의하고 있기 때문에 그 자체가 문서의 범위를 벗어나기 때문이다. DO-178B가 발간되기 전 1988년에 이미 FAA는 시스템 안전평가와 관련하여 FAR 25.1309 "Airworthiness Standard, Transport Category Airplanes Equipment Systems and Installations"[7]에 근거한 AC 25.1309-1A "System Design and Analysis"[8]를 발간하였으나 매우 제한적이며 실제 개발자들이 적용할 수 있을 만큼 포괄적이지 않았다. 또한 1996년에는 ARP 4754 "Certification Considerations for Highly Integrated or Complex Aircraft Systems"[9] 및 ARP 4761 "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment"[10]가 발간되었으나 두 문서 모두 강제 사항이 아니므로 이들 가이드라인의 적용에 대한 구체적인 피드백을 기대하기는 어렵다.

DO-178B와 대비되는 DS 00-55의 특징은 정형기법(Formal Methods)과 프로그램 분석(Static Analysis)을 강조한 표준이라는 점이다. DS 00-55 역시 안전 필수 소프트웨어에 초점을 맞추고 있기 때문에 고도의 완전성을 갖춘 소프트웨어 개발 프로세스를 지원하기 위해 이 두 가지 방법론을 요구해왔다. 그에 비해 DO-178B는 정형기법이나 프로그램 분석 기법을 부가적인 방법(Alternative Methods)으로서 간략히 언급하고 있다.

그런데 2004년 DS 00-56 3판이 발간되고 이어 2005년 DS 00-55가 폐기되면서 영국 국방성의 안전 소프트웨어 인증 체계는 구체적인 정형기법 중심의 가이드라인을 포기하고 목표 중심(Goal-Based)의 인증 기준을 제시하는 방향으로 선회한다. 이로써 프로세스 기반의 구체적인 가이드라인은 상실되고 소프트웨어 개발에 실질적으로 적용할 수 있는 세부 활동항목들은 더 이상 찾아볼 수 없게 되었다. 따라서 본 논문에서 다루고 있는 DO-178B와 DS 00-55의 비교는 시기적으로 실효성을 상실했다고 볼 수도 있다. 그러나 최근에 와서 구체적 활동 가이드라인의 부족이 다시 문제화되고 있는 만큼 두 표준간의 비교는 프로세스 활동항목의 구체화라는 측면에서 의미를 가진다.

나. DS 00-55의 정형기법 요구 활동

본 절에서 기술할 내용은 DS 00-55의 Part 1 (Requirements) 본문 분석을 통해 식별된 정형적(formal) 프로세스 활동 항목들이다. 즉, DS 00-55가 명시적으로 정형기법 또는 정적분석 기법을 요구하는 활동들을 SRS 개발 프로세스 측면에서 분류한 것이다. 분석은 DS 00-55 Part 1 본문 전체를 대상으로 실시하였으며 "Static Analysis", "Formal", "Formal Arguments", "Formal Representation", "Formal Specification", "Formal Verification" 또는 "Formal Methods"가 구체적으로 기술된 항목들을 식별하고 이 항목들과 관련된 프로세스 활동을 추출하는 방법으로 진행하였다.

표 5와 같이 DS 00-55는 소프트웨어 개발 과정 전반에 걸친 정형기법의 적용을 요구사항으로 제시하고 있다. 이 작업을 통해 본 논문이 목표로 하는 안전 필수 소프트웨어 개발 및 인증 기준 즉, DO-178에 필요한 정형기법 적용활동들을 파악함과 동시에 다음 장에서 소개할 CC EAL 7과 DO-178B가 수용가능한 정형적 활동 간의 매핑 작업을 위한 고려사항으로 활용한다.

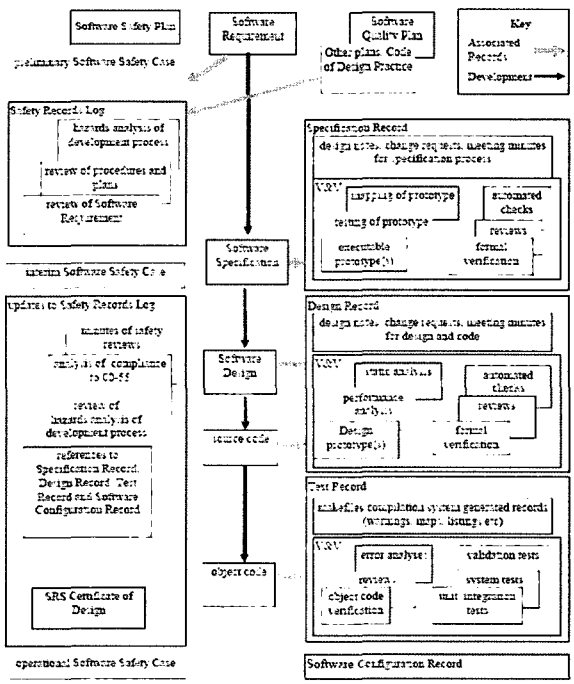
다. DS 00-55와 DO-178B의 프로세스 공유 가능성

DS 00-55와 DO-178B의 프로세스를 비교해보면 계획(Planning), 개발(Development), 통합(Integration)

[표 5] MoD DS 00-55 개발 프로세스 별 정형기법 적용 항목

구 분	정형기법 적용 항목
Planning	안전관리규정(Safety Case)
	안전성 입증 인자 (Safety Arguments)
	확인검증팀(V&V Team) 임무 정의
	방법론 선정(Selection of Methods)
	개발 언어 선정 (Selection of Language)
Development Principles	소프트웨어 명세, 설계 및 코드 생성 (Production of the SW Specification, SW Design and Code)
	추적성(Traceability) 유지
	소프트웨어 명세, 설계 및 코드 검증 (Verification of the SW Specification, SW Design and Code)
	정형 검증 리뷰(Review of formal verification)
Specification Process	예비 확인(Preliminary validation)
	정형 입증 인자를 통한 예비 확인 (Preliminary validation via formal arguments)
	실행 프로토타입을 통한 예비 확인 (Preliminary validation via executable prototype)
	명세 프로세스 점검 및 리뷰 (Checking and review of the specification process)
Coding Process	코딩 표준(Coding Standards)
	정적 분석 및 정형 검증(Static analysis and formal verification)
	오브젝트 코드 검증 (Object code verification)
	코딩 프로세스 점검 및 리뷰 (Checking and review of the coding process)
Testing and Integration	테스팅 원칙(Principles of testing)
In-Service	변경통제(Change control)

차원에서 상호 공유할 수 있는 항목들이 다수 존재한다. 표 5가 보여주는 프로세스 구분별 정형기법 적용 항목은 DS 00-55에 대한 분석 자료이지만 정형기법의 직접적인 언급을 제외하면 DO-178B에서도 대부분 다루어지는 것들이다. 따라서 DO-178B에서 이들 활동항목에 대한 정형기법의 적용원칙을 문서화하는데 구조적으로는 문제가 없다. 중요한 점은 단순하게 특정 활동에 대해 정형기법을 적용하라는 규정만으로는 부족하며 그 증거자료가 체계적으로 관리될 수 있는 구체적 생산물이 존재해야 하며 이들을 인증 증빙자료로 요청해야 한다는 것이다. 이런 관점에서 볼 때 DO-178B의 소프트웨어 개발프로세스 레코드는 매우 체계적인 자료 관리의 흐름을 보여준다. 그림 1에 나타난 프로세스 레코드 생성 흐름을 보면 소프트웨어 요구사항, 명세, 설계 및 코딩 프로세스가 진행되는 과정에서 명세 레코드, 설계 레코드, 테스트 레코드에 포함되어야 할 구체적인 자료들이 명시되어 있다. 특히 각 프로세스 별로 확인 검증 레코드를 구체적으로 요구하기 때문에 이러한 레코드가 수반되는



[그림 1] DS 00-55의 소프트웨어 개발 프로세스 레코드

프로세스 수행 결과에 대해 그 정확성과 완전성을 인증 받을 가능성은 매우 높다.

5장에서는 이와 같이 소프트웨어 수명주기 데이터와 직접적으로 관련된 프로세스 활동을 중심으로 DO-178B와 CC의 기준들을 비교한다.

5. DO-178B 수명주기데이터의 정형화

이 연구의 목표는 정형기법을 기반으로 DO-178B의 가이드라인을 보완하고 안전성과 보안성 인증 기준을 동시에 충족할 수 있는 기준으로 활용하고자 하는 것이다. 그에 따라 지금까지 안전성 중심의 분석을 통해 DO-178B의 취약성을 식별하였고, DS 00-55의 정형적 활동을 모델로 DO-178B의 프로세스 활동에 대한 정형기법 요구사항을 추출하였다. 지금부터는 DO-178B와 보안성 인증 기준인 CC, 특히 정형검증이 필수적인 EAL7과의 매핑을 통해 안전성과 보안성 인증 기준의 통합과정을 제시한다.

가. DO-178B vs. CC 비교 대상 및 범위

현실적으로 DO-178 문서의 모든 문장을 CC EAL 문서의 문장들과 일대일 대응시키기는 불가능하고, 그 목적이나 범위도 다른 부분이 있으므로 CC에서 정형기법을 요구하는 개발 클래스만(ADV)을 대상으로 그에 해당하는 DO-178B의 활동을 식별하기로 한다. 그리고 그 결과로서 DO-178B 11장 "수명주기데이터"의 각 항목에 대한 정형기법 적용여부를 결정하는데, 이는 수명주기데이터를 해당 수명주기의 최종 산출물로 볼 때 이와 관련된 일련의 활동들은 쉽게 식별되기 때문이다.

표 6에 나타난 바와 같이 CC EAL7의 경우 ADV_IMP와 ADV_INT를 제외한 모든 과정은 정형기법을 적용하여 수행해야 한다. 본 장에서는 정형기법을 요구하는 CC의 개발 클래스와 DO-178B의 수명주기 프로세스의 관계를 분석하고 DO-178B의 수명주기데이터 상에서 매핑 가능한 항목들을 식별한다.

나. DO-178B 수명주기 데이터의 중요성

DO-178B의 11장에서는 수명주기 프로세스의 각 활동에 대한 수명주기 데이터를 정의하고 있으며 그 개발 요건을 다음과 같이 규정하고 있다.

[표 6] CC EAL7 개발 클래스의 정형기법 요구수준 (CC v.2.3 기준)

CC Assurance Class	Assurance Family	Formal Methods in EAL 7
Development	ADV_FSP	Formal
	ADV_HLD	Formal
	ADV_IMP	•
	ADV_INT	•
	ADV_LLD	Semi-Formal
	ADV_RCR	Formal
	ADV_SPM	Formal

- Unambiguous : single interpretation
- Complete : all defined
- Verifiable : for correctness
- Consistent : no conflict
- Modifiable : changes in structure
- Traceable : to origin
- Form : for efficient retrieval
- Control : CC2 minimum

즉, 안전성 인증을 목표로 하는 수명주기 데이터는 모호함이 없어야 하고 모든 항목들이 완전하게 정의되어야 하며 그 정확성을 검증할 수 있어야 한다. 데이터 내부 혹은 다른 데이터와의 관계에서 상충되는 항목이 있어서는 안되며 쉽고 일관성 있게 수정 가능한 구조로 작성되어야 한다. 또한 원래의 요구사항 및 구현, 검증 결과 간의 추적성이 보장되어야 하며 재생산이 용이하도록 일정한 형식을 갖추어야 하고 최소한 Control Category 2에 해당하는 형상통제를 받아야 한다.

위와 같은 요건들을 충족하는 데이터를 작성하기 위해서는 언어의 정의가 명확해야 할 뿐만 아니라 지원 도구가 필수적으로 수반되어야 한다. 자연어 기술만으로는 데이터의 효과적인 관리뿐만 아니라 요구되

는 속성의 충족 여부 입증 자체가 어려울 수 있으며 실질적으로 더 많은 비용이 소모될 수 있다. 대표적인 예로서 "Verifiable(검증 가능해야 함)" 조건을 들 수 있는데 자연어 기술만으로 어떤 속성의 검증 가능성을 입증하기는 어려우며 이 경우 증빙 자료에는 막대한 분량의 테스팅 결과 혹은 분석결과들이 포함되어야 할 것이다. 이에 비해 정형언어를 사용한 명세와 정형기법을 기반으로 한 개발, 검증과정을 수행했다면 그 개발과정의 산출물 역시 정형 언어로 작성될 수 있을 것이고, 이 때 위와 같은 수명주기 데이터의 요건들은 특별한 부가 작업 없이도 충족시킬 수 있을 것이다.

다. DO-178B 수명주기 데이터와 CC 개발 클래스의 매핑

표 7은 본 연구에서 시도한 CC 본문의 개발 클래스들과 DO-178B의 수명주기 데이터 기술 항목과의 매핑 결과이다. CC에서 정형기법이 실질적으로 요구되는 보증 클래스는 ADV 즉, Development클래스이기 때문에 ADV 클래스만을 범위에 포함하였다. 또한 표 6에 나타난 바와 같이 ADV 중에서도 IMP 및 INT는 정형기법 적용 여부를 명확히 규정하기 힘들므로 분석 대상에서 제외하였고 이들을 제외한 모든 ADV 활동들을 정형기법 적용 항목으로 보고 매핑을 수행하였다.

실제로는 CC의 버전에 따라 EAL7에서 규정하는 정형기법 적용클래스가 다소 차이가 난다. 버전 2.3의 경우 ADV_FSP가 Formal, ADV_LLD가 Semi-formal 항목으로 지정되어 있으나 3.0(draft)의 경우 ADV_FSP가 Semi-Formal, ADV_LLD가 Formal 항목으로 지정되어 있으며 현재 3.0이 확정 공표되지 않은 상태에서 3.1이 발표될 것으로 알려지고 있다. 그러나 Formal과 Semi-formal 사이에서 약간의 변동이 있기는 하지만 EAL에 따른 보안등급 평가를 위해 어떠한 형태이든 정형적으로 기술되어야 할 부분이므로 이들을 정형기법 적용항목으로 간주한다.

본 연구에서 매핑이라 함은 CC 본문의 기술내용 분석을 통해 정형기법 적용 혹은 정형적으로 기술되어야 하는 부분을 먼저 식별하고 이 때 CC의 활동 항목에 대응할만한 DO-178B의 개발 프로세스 혹은

[표 7] CC EAL7 개발 클래스와 DO-178B 매핑 결과

Action Element	CC Paragraph	DO-178B Paragraph	Remarks
Functional Specification	ADV_FSP.2.1C	11.6	SW Requirement Standard
	ADV_FSP.2.1D	11.9	SW Requirement Data
	ADV_FSP.2.2C	11.9/ 11.14	SW Verification Results
	ADV_FSP.2.3C	11.9	SW Requirement Data
	ADV_FSP.2.4C	11.14	SW Verification Results
	ADV_FSP.2.5.C	11.9	SW Requirement Data
High-Level Design	ADV_HLD.2.1D	11.10	Design Description
	ADV_HLD.2.2C	11.14	SW Verification Results
	ADV_HLD.2.3C	11.7	SW Design Standard
	ADV_HLD.2.4C	11.10	Design Description
	ADV_HLD.2.5C	11.10	Design Description
	ADV_HLD.2.6C	11.10	Design Description
	ADV_HLD.2.7C	11.10	Design Description
	ADV_HLD.2.8C	11.10	Design Description
	ADV_HLD.2.9C	11.7	SW Design Standard
Low-Level Design	ADV_LLD.1.1D	11.10	Design Description
	ADV_LLD.1.2C	11.14	SW Verification Results
	ADV_LLD.1.3C	11.7	SW Design Standard
	ADV_LLD.1.4C	11.10	Design Description
	ADV_LLD.1.5C	11.10	Design Description
	ADV_LLD.1..6C	11.10a	Design Description
	ADV_LLD.1.7C	11.10c	Design Description
	ADV_LLD.1.8C	11.10c	Design Description
	ADV_LLD.1.9C	11.10c	Design Description
	ADV_LLD.1.10C	11.7	SW Design Standard
Representation Correspondence	ADV_RCR.2.1C	11.14	SW Verification Results
	ADV_RCR.2.1D	11.14	SW Verification Results
	ADV_RCR.2.2C	11.14	SW Verification Results
Security/ Safety Policy Modeling	ADV_SPM.3.1C	11.7	SW Design Standard
	ADV_SPM.3.1D	11.9	SW Requirement Data
	ADV_SPM.3.2C	11.14	SW Verification Results
	ADV_SPM.3.2D	11.14	SW Verification Results
	ADV_SPM.3.2C	11.9	SW Requirement Data

수명주기 데이터 항목을 대응시키는 것을 말한다. 예를 들어 CC의 기능 명세(Functional Specification) 활동 중ADV_FSP.2.1의 기능 명세에 대한 정형적 표현요구는 DO-178B의 11.6 Software Requirement Standard에서도 필요한 요구사항이라는 것이다.

라. 정형기법 적용 가능 수명주기 데이터 항목

위와 같이 CC의 ADV 보증 클래스 전체 항목에 대한 매핑을 수행하였으며 그 결과 정형기법 적용이 요구되는 DO-178B의 수명주기 데이터는 다음의 표 8에 나타난 바와 같이 6개 분야로 요약할 수 있다.

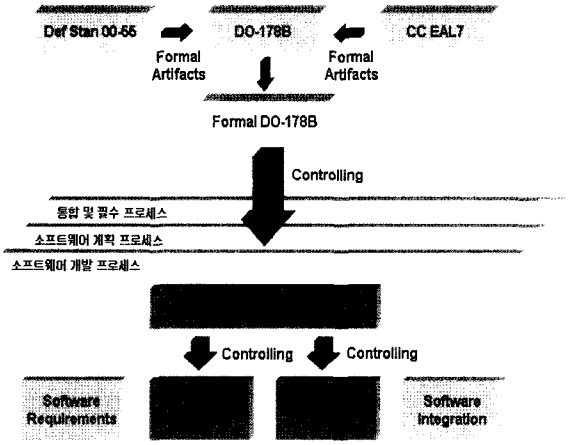
6. Ravenscar Profile

4장과 5장에서 다룬 표준과는 다소 다른 수준의 논의가 되겠지만 소프트웨어의 안전성 확보라는 측면에서 또 한 가지 고려할 수 있는 가이드라인으로서 Ravenscar Profile[11]을 들 수 있다. Ravenscar Profile은 항공기의 내장형 소프트웨어나 원자력발전소의 원자로와 같이 고도의 완전성(High-Integrity)이 요구되는 애플리케이션을 위한 실시간 태스킹 모델에 대한 정의 및 개발 가이드라인을 제공한다. Ravenscar Profile은 이러한 실시간 시스템이 요구하는 결정성(Determinism), 스케줄링 분석(Schedulability Analysis) 및 메모리 제한(Memory Boundedness) 등의 조건을 충족시키기 위해 정의된 Ada 태스킹 모델의 서브셋(Subset)이다. 정의에서 알 수 있듯이 Ravenscar Profile은 Ada 언어를 대상으로 개발된 가이드라인이지만 그 원칙이 명료하고 실시간 요구사항들이 매우 실질적으로 규정되었기 때문에 실시간 소프트웨어 개발을 위한 일반적인 가이드라인으로서도 충분히 의미를 가진다. 그 예로서 Real-Time Java를 위한 Ravenscar Profile의 적용 연구가 이미 수행되기도 하였다[12].

DO-178B나 CC가 소프트웨어 수명주기 전체를 High-Level에서 다루는데 비해 Ravenscar Profile은 소프트웨어의 설계, 구현 및 분석에 대한 직접적이고 구체적인 원칙을 제공한다. 또 한 가지 중요한 Ravenscar Profile의 철학은 정형적 분석이 가능하도

[표 8] DO-178B 수명주기 데이터의 정형기법 적용 항목

수명주기 데이터	세 부 항 목
Software Requirements Standards	▪ 소프트웨어 요구사항 개발 방법론 ▪ 요구사항 표기 방법 - 정형명세 언어 ▪ 요구사항 개발 도구 ▪ 시스템 프로세스에서 추출된 요구사항
Software Design Standards	▪ 설계 표기 방법론 ▪ Naming convention ▪ 설계 방법론 적용 고려사항 ▪ 설계 도구 사용 고려사항 ▪ 설계 고려사항 ▪ 복잡도 고려사항
Software Requirements Data	▪ High-Level 요구사항에 대한 정의 ▪ 시스템 - SW 요구사항간 할당 관계 ▪ 기능 및 운영 요구사항 ▪ 성능 기준(정밀도 및 정확도) ▪ 시간 요구사항 및 시간 제약사항 ▪ 메모리 용량 ▪ HW-SW 인터페이스(프로토콜, 포맷, 입출력 주기) ▪ 실패 감지 및 안전성 감시 요구사항 ▪ 요구사항의 분할
Design Description	▪ SW Low-Level 요구사항에 대한 정의 ▪ SW High-Level 요구사항 충족 여부 ▪ SW 아키텍처: 소프트웨어 구조 정의 ▪ 입·출력 ▪ 데이터/컨트롤 플로우 ▪ 자원 제약 사항 및 자원 관리 정책 ▪ 스케줄링 및 프로세스/태스크 간 통신 방식 ▪ 설계 방법론 및 구현 세부 항목 ▪ 분할 방법 ▪ 신규 혹은 기 개발 컴포넌트 속성 ▪ 설계 수행 중 도출된 추가 요구사항 ▪ Deactivated 코드 ▪ 설계와 안전관련 요구사항과의 추적성
SW Verification Cases and Procedures	▪ 리뷰 및 분석 프로시저 ▪ 테스트 케이스 ▪ 테스트 프로시저
Software Verification Results	▪ 리뷰, 분석 및 테스트 결과 ▪ 리뷰, 분석 및 테스트된 형상 항목 또는 소프트웨어 버전



[그림 2] 안전/보안필수 소프트웨어 개발 프로세스 통제 개념

록 소프트웨어를 설계하고 구현해야 한다는 것이다. 이 점에서 Ravenscar Profile은 본 연구가 추구하는 정형적 개발/검증 가이드라인 구축에 있어 소프트웨어 작성과 직접적으로 관련된 세부 원칙을 제공할 수 있다.

4, 5장에서 제시한 정형적 표준과 Ravenscar Profile의 연관성을 정리하면 그림 2와 같다. 즉 본 연구가 목표로 하는 정형기법 적용 DO-178B는 프로그램의 계획(Planning), 개발(Development) 및 필수(Integral) 프로세스를 전반적인 차원에서 통제하며 Ravenscar Profile은 개발 프로세스 내에서 수행되는 설계 및 코딩활동에 대한 세부 가이드라인을 제공하는 것이다. 그림에서는 나타나지 않지만 Ravenscar Profile을 충실히 수행한 개발, 특히 Ada 프로그램이라면 매우 결정적(Deterministic) 인 동작 속성을 가지게 되고 이는 곧 정형 검증 및 분석에도 적합하게 될 것이다. 따라서 이 경우에는 필수 프로세스의 검증 활동과도 연관된다고 볼 수 있다.

7. Case Study : 정형기법 적용 사례

정형기법의 적용은 안전성 또는 보안성 인증 목적의 소프트웨어 개발에 있어 두 가지 측면의 문제를 동시에 해결해준다. 먼저 개발자 혹은 피인증자 측면에서 문제시 되는 안전성 분석 및 검증 가이드라인의 부족 문제이다. 위와 같이 정형기법이 적용되어야 할 해당 패러그래프의 데이터를 명시하고 정형기법 적용에 대한 증빙 자료를 요구하는 경우 피인증자가 수행해야 할 활동방향과 그 산출물이 명확해진다. Statechart, Z를 비롯하여 현재 정형언어로 인정되는 언어들이 해당 데이터의 기술 수단이 될 수 있으며 정형언어를 사용한 설계와 검증의 결과는 모호함과 불완전성이 배제됨을 전제하고 있으므로 인증 타당성에 대한 논란의 소지를 제거할 수 있다.

가. 정형 명세 및 검증

이 절에서 소개되는 연구결과는 고도의 완전성을 요구하는 Ada Ravenscar Profile[7]을 적용하여 작성된 Ada 코드에 대하여 역공학적 방법을 동원하여 동일한 행위를 수행하는 Statechart로 변환한 예이다.

다음은 [11]에서 소개된 4개의 Ada 태스크와 3개의 보호객체(Protected Object)로 이루어지는 소형 실시간 프로그램 대한 요구사항들이다. 본 절에서는 이 프로그램에 대해 Ravenscar Profile 기준을 충족하는 Statechart 정형명세 및 정형검증 결과를 제시한다.

1. *Regular Producer는 주기적 태스크를 포함하고 있으며 Small Whetstone 프로그램으로부터 일정 크기의 업무를 할당 받아 수행한다.*
2. *Regular Producer는 On Call Producer의 업무로드 기준치가 초과한 경우, 초과한 On Call Producer의 업무로드를 인수하여 수행한다.*
3. *On Call Producer는 2에 의해 동작하는 산발적 태스크를 포함한다.*
4. *Regular Producer의 요청에 의하여 On Call Producer의 Start 오퍼레이션이 수행되며 이 때 Start 오퍼레이션은 Request Buffer의 Deposit 오퍼레이션을 통해 요청 데이터를Request Buffer에 저장한다. Request Buffer는 이전 요청을 수행하는 도중에도 새로운 요청을 수용할 수 있도록 보호객체로 선언되어야 한다.*
5. *시스템이 업무로드를 수행하는 동안 외부 장치로부터 산발적인 인터럽트가 발생할 수 있으며 그*

인터럽트는 보호객체 Even Queue의 Signal 오퍼레이션을 통해 전달된다.

6. *산발적 태스크인 External Event Server는 인터럽트의 개수를 기록하고 Activation Log의 Write 오퍼레이션을 통해 각 인터럽트의 발생 시간을 기록한다.*
7. *Regular Producer는 외부장치의 서비스 요청을 감시하기 위해, 지정된 조건을 만족하는 경우 Activation Log Reader의 Read 오퍼레이션을 통해 Activation Log의 기록을 읽는다.*
8. *Regular Producer는 Suspension 객체를 통해 태스크간 동기화를 수행한다.*

이상의 자연어 요구사항을 충족하면서 요구사항 간 서로 상충하지 않는 모델을 설계하기 위해 본 연구에서는 Statemate Magnum(v.4.1)[13]의 Statechart를 사용하였다. 실제 연구과정에서 사용된 소스코드 분석 및 역공학적 해석 방법은 본문에서 구체적으로 소개하지 않는다. 대신 이 절에서는 자연어 요구사항과 그에 대한 정형 명세 혹은 정형 설계의 결과물을 비교함으로써 정형적 표현방식의 적용 가능성을 보이고자 한다.

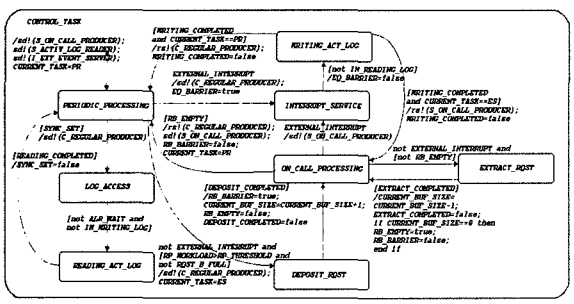
그림 3은 예제 시스템에 대한 위의 8가지 요구사항을 반영하여 작성한 Statechart이다. 이 차트 내부에는 각 태스크들의 동작 상태와 보호객체에 대한 읽기 상태, 그리고 쓰기 상태 등을 각각 하나의 State로 정의하였다. 여기서 태스크의 상태를 좀 더 구체적으로 말하면 태스크 자체의 ON/OFF 상태가 아니라 태스크의 고유 기능 즉, 태스크의 함수나 프로시저가 수행되는 상태이다. 이는 Ravenscar Profile에서 태스크가 종료하지 않는 것으로 정의되어 있으므로 항상 동작(ON) 상태를 유지한다고 전제하고 태스크의 우선순위에 따라 자신의 업무로드를 실행하는 것, 즉, 순수하게 기능적인 동작상태만을 표현했기 때문이다.

각각의 State와 함께 State 간의 전이 조건 혹은 전이 결과를 명시함으로써 실행 가능한(Executable) 명세가 Statechart 모델로 작성되었으며 Statemate 자체 도구의 지원을 받아 모델 내부의 정확성(Correctness)과 완전성(Completeness)을 점검할 수 있었다. 또한 Statemate에 Plug-in된 정형검증 도구인 ModelChecker를 사용하여 모델 내부의 비결정성(Non-Determinism)과 Race Condition, 데이터 범위 침범 및 특정 상태에 대한 도달 가능성 등을 검증할 수 있었다[14]. 본 절의 내용과 관련된 구체적인 연구 결과는 [14]에 잘 나타나 있다.

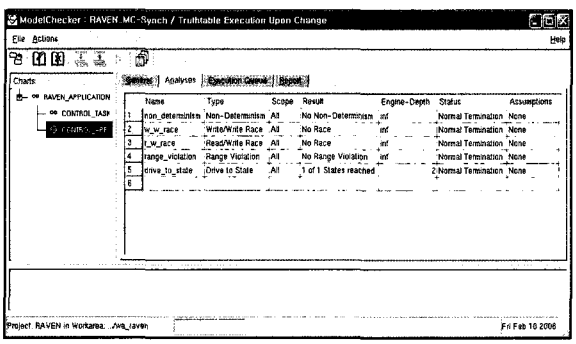
그림 4는 그림 3의 Statechart에 대해 Non-Determinism, Write-Write Race, Read-Write Race, Range Violation, Drive to State 등의 속성에 대해 ModelChecker를 수행한 결과이다. 이와 같이 실행 가능한 정형명세와 정형검증 결과는 소프트웨어의 안전성 인증에 있어 피인증자 측의 중요한 증빙자료가 될 뿐만 아니라 인증자 측면에서도 명확한 평가 기준으로 사용될 수 있다.

나. 컴포넌트의 재사용

정형기법 적용의 또 하나의 이점은 정형성을 기반으로 인증 받은 제품 혹은 소프트웨어 컴포넌트의 재사용 가능성 확대이다.



[그림 3] Statechart를 이용한 시스템 행위 모델의 설계

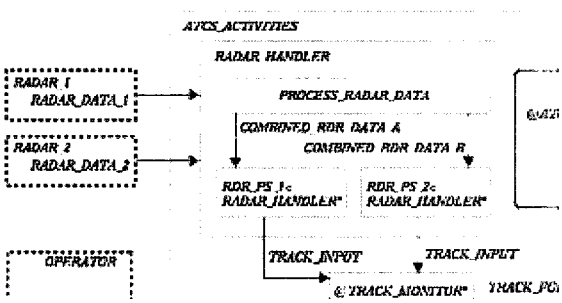


[그림 4] Statemate ModelChecker 수행 결과

그림 5는 Statemate Magnum으로 작성된 항공관제시스템의 예로서 레이더 입력 모듈의 재사용이 가능하도록 "Generic Chart"로 작성한 것이다. 그림의 RADAR HANDLER는 시스템 외부 레이더로부터 입력을 받아 시스템이 처리할 수 있는 데이터 포맷으로 전환시키는 모듈이다. 만약 이 모듈이 재사용을 고려하지 않고 특정 레이더 포맷만을 처리할 수 있도록 설계되었다면 다수의 외부 레이더의 전송 데이터 포맷이 서로 다를 경우 필연적으로 각각의 레이더 입력을 처리하기 위해 다른 모듈을 추가적으로 설계해야 한다.

그러나 어떠한 레이더 입력포맷에 대해서도 처리할 수 있는 모듈을 설계하고 그 기능 및 안전성을 완벽하게 검증했다면 이 모듈을 재사용 할 경우에도 그 기능과 안전성을 보장받을 수 있을 것이다. 물론 이와 같이 특정 소프트웨어 모듈을 재사용하는 경우에는 테스팅에 의한 검증이 더욱 어렵다. 아직 존재하지도 않은 레이더 입력 포맷에 대해서도 검증을 수행해야 할 지 모르기 때문이다. 다행히 Statemate의 Generic Chart 내부에는 Statechart를 포함시킬 수 있어 모듈의 행위에 대한 정확성과 안전성을 모델체킹을 통해 검증할 수 있다[15]. 그림 5와 관련된 구체적인 설계 및 검증 과정은 [15]에서 확인할 수 있다.

재사용 컴포넌트의 기능성과 안전성, 그리고 Generic Chart나 LynxOS-178의 예에서 볼 수 있는 독립성이 보장될 경우 재사용 컴포넌트의 인증은 곧 이 컴포넌트의 재사용에 대한 재인증의 필요성을 줄일 수 있다.



[그림 5] Generic Chart를 사용한 재사용 모듈의 설계

8. 결 론

소프트웨어 보안성 및 안전성 관련 표준들이 Goal-Based화 되고 있는 배경에는 소프트웨어 개발 기술의 진보와 신뢰할 수 있는 개발도구들의 등장 그리고 개발의 효율성을 추구하는 산업계의 움직임이 작용하고 있다. 본 논문에서는 Goal-Based 규제로 인한 구체적 개발 및 검증 가이드라인의 부족을 DO-178B의 취약점으로 거론하였지만 다른 한편에서는 DS 00-55의 사례와 같이 표준의 세부적 규제 항목들을 통폐합하여 Goal-Based 표준으로 전향하려는 움직임 역시 존재한다. 다양하고 효율적인 개발 도구 또는 방법론의 적용을 Goal-Based 표준의 장점으로 들 수도 있고, 인증 권한자 측면에서도 목표 충족 여부를 결과론적으로 확인하면 되기 때문에 그 프로세스가 단순명료하다고 볼 수도 있다. 그러나 본 연구에서 설정한 DO-178B의 문제점은 바로 그러한 Goal-Based적 특성에서 기인한 것들이 많다. 이들을 다시 정리하면 다음과 같다.

- 목표 해석의 다양성
- 검증되지 않은 방법론의 적용 가능성
- 인증 증빙자료의 난립 가능성

즉, 구체적 활동 가이드라인의 부재에서 오는 개발자들의 임의 해석이나 독자적인 방법론 적용에 의한 위험성뿐만 아니라 모호한 검증 결과를 놓고 설득력 있는 자료로 포장하기 위해 무분별한 자료들이 오용될 가능성 또한 크다는 것이다.

본 연구는 그 범위가 소프트웨어에 한정되고 특히 일반적인 수명주기 프로세스 중에서도 개발 프로세스에 국한되지만 정형기법의 적용을 통해 개발 활동 수행상의 오류를 최소화 하고 그 산출물을 검증 가능한 형태로 유도하고자 하였다. 소프트웨어 안전성 및 보안성 인증 기준 충족을 위한 본 연구의 목표를 요약하면 다음과 같다.

- 인증 목표 충족을 위해 요구되는 개발 활동의 구체적 명시
- 수명주기 데이터(산출물)의 정형적 기술

- 요구사항 및 설계에 대한 정형검증 가능성 확인
- 기 인증 항목에 대한 재사용 가능성 확인

그 결과, DO-178B의 개발 프로세스를 기준으로 볼 때, 소프트웨어 요구사항 작성 및 설계 방법론, 요구사항 명세, 설계 명세, 검증 계획 및 절차, 검증 결과 등에서 타 표준에서 이미 적용했거나 적용 중인 정형기법을 수용할 수 있다고 판단하였다. 그리고 CC와의 통합 과정에서 확인한 바와 같이 이러한 프로세스 활동은 안전성뿐만 아니라 보안성 요건에도 공통적으로 적용 가능한 것으로 판단되며 그 최종 목표는 소프트웨어의 완전성(Integrity) 향상에 있다. 그리고 정형기법에도 많은 표기방법과 다양한 지원도구들이 존재하지만 그 다양성에 관계없이 정형기법의 가장 큰 장점은 정형적 기술 내용을 해석함에 있어 모호함이나 충돌이 발생할 가능성이 매우 적다는 것이다. 그러므로 Z, Statechart 등 정형언어로 인정되는 표기방법, 그리고 Model Checking, Theorem Proving 등으로 대표되는 정형검증 방법이 올바르게 사용되었다면 그 기법의 다양성은 크게 문제되지 않는다.

본 연구가 제시하는 정형기법 적용을 통해 수정된 DO-178B의 개발 가이드라인은 소프트웨어 계획, 개발 및 필수(Integral) 프로세스를 통제하면서 각 프로세스 활동에 대하여 정형적으로 기술된 산출물 즉, 수명주기 데이터들을 요구한다. 그리고 개발 프로세스의 경우 보다 구체적으로 그 설계와 구현 과정에 Ravenscar Profile이 제공하는 실시간 속성과 안전성 확보 정책을 적용함으로써 소프트웨어의 완전성(Integrity)을 향상시키고자 하였다.

정형기법 적용의 실효성에 대해서는 Case Study를 통해 간접적으로 제시하였다. 즉, Statechart 기반의 검증 및 재사용 컴포넌트 설계연구에서 보였듯이 편리한 지원도구와 명료한 표기 방법을 통해 일반적으로 인식되는 정형기법의 난해함을 극복할 수 있으며, 설계 및 검증 결과의 재사용 가능성 역시 증가됨을 알 수 있다.

참 고 문 헌

[1] RTCA, "DO-178B; Software Consideration in Airborne Systems and Equipment Certification", 1992.

[2] Common Criteria.

[3] UK MoD, "Def Stan 00-55; Requirements for Safety Related Software in Defense Equipment Issue 2", 1997.

[4] John A McDermid, "Trends in System Safety : A European View?", 7th Australian Workshop on Safety Critical Systems and Software, Adelaide, 2002.

[5] Carolyn Salmon, "The Certification of Systems containing Software Developed using RTCA DO-178B", ERA, 2006.

[6] Hoyt Lougee, "DO-178B Certified Software : A Formal Reuse Analysis Approach", The Journal of Defense Software Engineering, 2005.

[7] Federal Aviation Regulations 25.1309, "Airworthiness Standard, Transport Category Airplanes Equipment Systems and Installations", 1977.

[8] FAA Advisory Circular 25.1309-1A, "System Design and Analysis", 1988.

[9] ARP 4754, "Certification Considerations for Highly Integrated or Complex Aircraft Systems", 1997.

[10] ARP 4761, "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment", 1996.

[11] Alan Burns, Brian Dobbing and Tullio Vardanega, "Guide for the Use of the Ada Ravenscar Profile in High Integrity Systems", University of York Technical Report YCS-2003-348, 2003.

[12] Jagun Kwon, Andy Wellings, Steve King, "Ravenscar-Java : A High Integrity Profile

for Real-Time Java", York Technical Report YCS 342, 2002.
[13] David Harel, "Modeling Reactive Systems with Statecharts : The Statemate Approach", I-Logix, 1999.
[14] Chang-Jin Kim, Jin-Young Choi, "Transformation of the Ravenscar Profile Based Ada Real-time Application to the Verification-ready Statecharts : Reverse engineering and Statemate approach", SERP2006, 2006.
[15] 김창진, 최진영, "소프트웨어 설계 모듈의 재사용을 위한 Statemate 일반화 차트의 확장", 2006.