# HELISCOPE Project 의 비행 운용 프로그램을 위한 검증 절차*

이종훈, 이동아, 유준범

건국대학교 컴퓨터공학부
서울 광진구 화양동 1 번지 건국대학교
kirdess@konkuk.ac.kr, ldalove@konkuk.ac.kr, jbyoo@konkuk.ac.kr

**요약**: HELISCOPE Project 는 무인비행체 기술을 융합하여 재난 현장에 투입하여 활용할 수 있는 방재용 무인 헬리콥터 개발을 위한 연구이다. 무인 헬리콥터에는 비행제어컴퓨터가 탑재되어 동작하며, 비행제어컴퓨터에는 Operational Flight Program(OFP)이 탑재되어 각종 비행제어와 관련된 역할을 수행한다. 따라서 OFP 의 결함은 비행 상태와 직결되어 치명적인 피해를 가져올 수 있기 때문에 OFP 의 안전성을 보장하기 위한 검증 활동이 엄격하게 이루어져야 한다. 본 논문에서는 HELISCOPE Project 의 OFP 를 대상으로 안전성을 확보하기 위하여 항공 분야에서 표준으로 사용되고 있는 DO-178B 지침 방안과, IEEE 1012-2004 소프트웨어 검증과 확인 절차 표준을 참조한 정형화된 검증 절차를 제시한다. 본 검증 절차에는 OFP 가 개발이 완료된 소프트웨어인 점을 고려하여 소프트웨어 역공학 과정을 더하여 구성하였다. 소프트웨어 역공학 과정을 통하여 검증을 수행하기 위해 필요한 소프트웨어의 요구사항과 설계 정보를 보완하고 복구한다. 또한 OFP 가 실시간으로 동작하는 내장형 소프트웨어인 점을 고려하여 정적 코드 분석, 정형 검증, 시험 활동으로 구성한 검증 절차에 대한 내용을 기술한다. 본 논문에서 제시하는 소프트웨어 검증 절차를 수행함으로써 HELISCOPE Project 의 OFP 에 대한 안전성을 확보할 수 있을 것으로 기대한다.

**핵심어:** HELISCOPE Project, OFP(Operational Flight Program), OFP Verification Process, Formal Verification, Static Code Analysis, Software Testing

## 1. 서론

오늘날 소프트웨어 개발 산업은 크게 발달하여 많은 산업 분야에서 필요성이 매우 커지게 되었고, 일반 가정에서 사용하는 전자제품부터 거대한 산업 시스템까지 소프트웨어가 차지하는 영역과 비중 또한 커지게 되었다. 작은 결함이 큰 피해를 야기할 수 있는 산업 분야에서도 소프트웨어는 중요한 위치를 차지하고 있다. 따라서 이러한 분야에서 사용되는 소프트웨어는 안전성(Safety)을 확보하기 위한 체계적이고 정형적인 검증 절차가 필요하다. 특히 항공 분야는 높은 안전성이 요구되는 대표적인 분야이며 다른 산업 분야와 마찬가지로 소프트웨어가 큰 비중을 차지한다.

HELISCOPE Project [1]는 대형 재난의 예방과 복구를 위하여 빠른 정보수집과 높은 이동성을 가지는 소형 무인 헬리콥터 개발을 목표로 하는 연구이다. 무인 헬리콥터에는 비행제어컴퓨터(FCC: Flight Control Computer)가 탑재되어 있으며 비행제어의 핵심적인 부분을 담당하는 소프트웨어인 Operational Flight Program(OFP) [2]이 탑재된다. OFP 의 결함은 곧 비행제어의 이상으로 이어지며, 이는 임무의 실패뿐만 아니라 인명, 재산, 환경적인 피해를 가져올 수 있다. 따라서 OFP 는 높은 소프트웨어 안전성이 요구된다. 이러한 높은 안전성을 확보하기 위해서는 국제 표준에 의거한 검증 절차 수립 및 검증 수행이 이루어져야 한다.

IEEE 1012-2004 소프트웨어 검증과 확인 절차 (Software Verification & Validation Process) 표준 [3]은 소프트웨어 검증을 위한 절차를 제시하는 대표적인 표준이다. 항공 소프트웨어 분야에서는 미국 연방항공국(Federal Aviation Administration)에서 항공기에 사용되는 소프트웨어의 안전성을 확보하기 위해 DO-178B 개발 지침 [4]을 안전성 인증을 위한 용도로 사용할 것을 인정하여 상용 항공기 표준 인증으로 사용되고 있다. 이들 표준은 소프트웨어의 전체 개발 생명 주기에 대한 내용을 담고 있으며 다양한 형태의 소프트웨어에 적용하기 위해 내용이 다소 개괄적인 면이 있다. 따라서 검증할 대상에 따라 그 특성을 고려하여 절차를 구성해야 한다.

본 논문에서는 DO-178B 개발 지침과 IEEE 1012-2004 소프트웨어 검증과 확인 절차 표준을 참조하여 HELISCOPE Project 의 비행제어컴퓨터에 탑재되는

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음(NIPA-2010-C1090-0903-0004, NIPA-2010-C1090-1031-0003). 이 논문은 2010 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2010-000256).

OFP 를 대상으로 하는 정형화된 검증 절차를 제시한다. OFP 의 특징이 실시간으로 동작하는 내장형 소프트웨어(Embedded software)라는 점, 그리고 개발이 완료된 소프트웨어 라는 점을 고려하여 부족한 정보를 보완하고 복구하는 역공학(Reverse engineering) 과정과 정적 코드 분석(Static code analysis) 과정, 정형 검증(Formal verification) 과정, 시험(Test) 과정을 포함한 절차를 제시한다.

본 논문의 구성은 다음과 같다. 2 장에서는 소프트웨어 검증 관련 표준과 HELISCOPE Project 및 검증 절차와 관련된 연구에 대하여 살펴본다. 3 장에서는 OFP 에 대한 개요와 검증 시 고려할 특성에 대하여 정리한다. 4 장에서는 전체적인 검증 절차와 각 단계별 세부 활동을 제시하고, 5 장에서는 결론과 향후 연구 방향에 대하여 정리한다.

## 2. 관련 연구

### 2.1. 검증 절차 관련 표준

#### 2.1.1. IEEE 1012-2004

IEEE 1012-2004 는 소프트웨어 검증과 확인 절차에 대한 표준 문서이다. IEEE 1012-2004 표준은 개발 생명 주기에 걸쳐 진행되는 검증과 확인 활동 절차에 대한 내용을 담고 있으며 관리(Management), 획득(Acquisition), 공급(Supply), 개발(Development), 운영(Operation), 유지(Maintenance) 단계로 이루어져 있다. 위험성 분석, 시험 등의 검증 활동을 수행하도록 정의하고 있으며 검증 활동은 '개발'단계에서 시험 활동을 중심으로 이루어진다.

IEEE 1012-2004 에서는 위험성 분석(Criticality analysis)을 통한 소프트웨어 무결성 등급(SIL: Software Integrity Level)을 결정하고 있다. 무결성 등급은 Level 1 ~ 4 의 4 개의 등급으로 분류하며 등급에 따라 검증을 적용하는 강도가 달라진다.

#### 2.1.2. DO-178B

DO-178B 는 항공 소프트웨어 개발자들과 정부 검증 조직들을 위한 지침 방안으로, 미국 항공 무선 기술 위원회(RTCA: Radio Technical Commission for Aeronautics)의 연방 자문 위원회에 의해 개발 되었다. DO-178B 지침 방안은 항공 소프트웨어의 개발 생명 주기와 소프트웨어 인증에 대한 내용을 담고 있다. 전체 절차는 계획(Planning), 개발(Development), 검증(Verification), 형상관리(Configuration Management), 품질보증(Quality Assurance), 인증(Certification Liaison) 단계로 이루어져 있으며 소프트웨어 개발 외에도 품질을 보증하기 위해 전체 활동에 대한 검토와 감사, 인증에 대한 절차를 포함하고 있다. DO-178B 에서는 검토, 분석, 시험 등의 검증 활동을 수행하도록 정의하고 있으며 이는 '검증' 단계에서 이루어진다.

DO-178B 는 시스템 안전평가(System safety assessment)를 통해 소프트웨어의 등급(Software Level)을 결정하고 있다. 소프트웨어 등급은 Level A ~ E 의 5 개의 등급으로 분류하며 등급에 따라 검증을 적용하는 강도가 달라진다.

### 2.2. HELISCOPE Project

HELISCOPE Project [1]는 재난의 예방과 복구를 위하여 빠른 정보수집과 높은 이동성을 가지는 소형 무인 헬리콥터 개발을 목표로 하는 연구이다. 이 헬리콥터는 재난 발생 시 현장으로 빠르게 이동하고 현장에 대한 정보를 전송하기에 적합해야 한다. 이를 위해 비행제어컴퓨터와 멀티미디어통신컴퓨터(MCC: Multimedia Communication Computer)를 헬리콥터에 설치하여 내장형 시스템으로 구현하는 방법에 대한 연구가 진행되어 왔다.

HELISCOPE Project 의 OFP 는 비행제어컴퓨터에서 동작하는 내장형 소프트웨어로써, 여러 센서를 통하여 실시간으로 변화하는 헬리콥터의 상태를 확인하고, 안전하고 안정된 비행제어를 위해 다음 움직임에 대한 명령을 일정 시간 내에 계산할 수 있도록 한다. 또한 멀티미디어통신컴퓨터는 소형 카메라를 이용하여 촬영된 현장의 정지영상 및 동영상을 무선인터넷을 통해 지상관제시스템에 중계하여 재난 현장에 대한 정보를 확인할 수 있도록 한다. 지상관제시스템(GCS: Ground Control System)은 비행제어시스템과 통신하여 헬리콥터의 현재 상태를 확인할 수 있게 하고, 카메라에 대한 조작을 할 수 있게 한다. 이러한 사항들을 만족하기 위해 헬리콥터의 무인비행모드와 카메라의 동작모드를 정의하고 실시간 내장형 소프트웨어인 OFP 를 개발하는 연구가 진행되어 왔다.

### 2.3. 검증 절차 연구

비행제어시스템 설계 및 검증 절차 연구 [5] 는 T-50 초음속 고등훈련기급 항공기 개발 사례를 경험으로 비행제어시스템 설계 및 검증 절차(FCSDVP: Flight Control System Design and Verification Process) 모델을 제시하고 있다. 항공기의 조종성능, 안정성 그리고 비행안전성을 확보하기 위해 비행제어시스템의 설계와 비행시험에 대하여 언급하고 있으며, 비행시험의 일부는 OFP 를 검증하는 과정으로 구성되어 있다. OFP 의 모듈에 대한 단위시험을 수행하고, HILS 환경을 이용하여 OFP 의 기능, 조종사 입력에 대한 반응을 확인하고 하드웨어 동작을 검증한다.

높은 안전성이 요구되는 원자력 발전소의 계측제어계통 소프트웨어에 대한 시험을 수행하기 위한 연구 [6] 는 국내외 표준을 참조하여 시험 절차와 활동을 제시하였다. 소프트웨어의 등급을 안전-필수, 안

전-관련, 비안전의 3 등급으로 분류하여 등급별 시험 방법을 달리 적용하도록 하였다.

KNICS Project 를 위한 검증과 확인(V&V)절차 연구 [7] 는 원자력 발전소의 PLC 기반 ESF-CCS 시스템의 높은 성능과 신뢰성, 품질을 확보하기 위한 절차를 제시한다. 요구사항과 디자인에 대한 검사(Inspection)활동과 추적성 분석, 정형 검증을 수행하며, 구현 단계에서는 시험 활동을 주로 수행하는 것으로 정의하고 있다. 또한 안전성 분석(Safety analysis)과 형상 관리에 대한 내용도 정의하고 있다. 해당 연구에서는 NUREG-0800, RG 1.170, IEEE 7-4.3.2, IEEE 1012, IEEE 1028, IEEE 1008, IEEE 829 등의 표준을 참조하여 원자력 발전소 ESF-CCS 에 대한 체계적인 검증과 확인 절차를 제시하였다.

## 3. Operational Flight Program

### 3.1. 개 요

OFP 는 항공기에 탑재되어 동작하는 소프트웨어이다. 항공기의 제어를 담당하는 비행제어컴퓨터에는 실시간 운영체제(RTOS: Real Time Operating System)가 탑재되어 동작하며, OFP 는 이 비행제어컴퓨터에 탑재되어 동작하게 된다. OFP 는 지상관제시스템(GCS: Ground Control System)과 통신하며, 현재 항공기의 상태를 파악하고 안정적인 비행 상태를 유지하기 위한 제어 명령을 내리는 역할을 한다. 이를 위해 OFP 는 비행제어에 필요한 각종 제어법칙들을 포함하고 있다. 빠르게 변화하는 항공기의 상태를 파악하고 제어 명령을 내리기 위하여 OFP 는 실시간으로 동작해야 하는 특징을 가지며, 비행제어컴퓨터에 탑재되어 동작하는 내장형 소프트웨어의 특징을 가진다.

### 3.2. 특 징

3.2.1. 실시간 소프트웨어(Real Time Software)

HELISCOPE Project 의 OFP 의 주 역할과 기능은 센서들로부터 현재 상태에 대한 정보를 받아 이를 기반으로 다음 움직임에 대한 명령을 계산하여 각 서보 모터에 전달하고, 현재 상태를 지상관제시스템으로 전송하는 것이다. 따라서 안전하고 안정적인 비행 상태를 유지하기 위해 다음 움직임에 대한 명령을 일정 시간 내에 계산하여 전달해야 한다. HELISCOPE Project 에서는 비행운용제어소프트웨어의 다음 동작에 대한 계산이 20ms~50ms 시간 안에 이루어져야 한다고 정의 [1]하고 있으며, 이를 만족하기 위해 실시간 운영체제인 RT-eCos3.0 를 사용하고 TMO 모델 [8]을 적용하여 설계되고 개발되었다. 따라서 OFP 는 시간 조건에 매우 민감한 특징을 보이며, 안전성을 확보하기 위해 이러한 특징이 고려되어야 한다.

3.2.2. 내장형 소프트웨어(Embedded Software)

HELISCOPE Project 의 OFP 는 내장형 소프트웨어이기 때문에 하드웨어적인 제약사항이 있다. 낮은 처리 성능, 작은 메모리 영역 등에 의해 성능적인 문제가 발생할 수 있으며, 실제 환경에서의 시험 수행이 어려워질 수 있다. TMO 모델을 적용하여 개발되었기 때문에 센서의 입력에 따라 동작하는 특징(Event-driven)과, 공유데이터 영역을 사용하며 이에 접근하는 모듈간 상호배제가 이루어져야 하는 특징도 가지고 있다. 따라서 안전성을 확보하기 위해 이들 특징이 고려되어야 한다.

## 4. 검증 절차

IEEE 1012-2004 표준과 DO-178B 지침에서는 소프트웨어의 개발 생명 주기 전체에 대한 활동을 정의하고 있다. 그러나 본 연구에서 대상으로 하는 OFP 는 이미 개발이 완료된 소프트웨어로서 개발 생명 주기의 대부분이 이미 진행된 상태이다. 따라서 생명 주기 각 단계에 해당하는 산출물인 요구사항과 설계에 관한 정보가 불완전하거나 누락되어 있을 수 있으며, 표준에서 요구하는 산출물의 수준에 비해 내용이 부족할 수 있다. 또한 OFP 의 안전성을 좀 더 면밀히 검증하기 위해 3.2. 에서 언급한 특징들을 고려한 활동들이 절차에 포함되어야 한다.

본 절차에서는 IEEE 1012-2004 표준과 DO-178B 의 내용을 수정하여 구성한 검증 절차를 제시한다. 개발 생명 주기의 요구사항과 설계 단계의 산출물의 보완 방안으로 소프트웨어 역공학 과정을 구성하였다. 역공학 과정을 통하여 검증 활동을 수행할 정보를 보완하고 복구할 수 있다. OFP 의 특징들을 고려한 검증을 수행하기 위한 방안으로 정적 코드 분석, 정형 검증, 시험 활동을 절차에 포함시켜 구성한다. 그림 1 은 전체적인 검증 절차의 흐름을 나타낸 것이다.

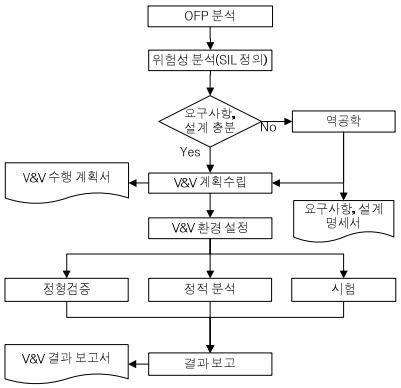


**그림 1 OFP 검증 절차**

### 4.1. 소프트웨어 등급 정의

IEEE 1012-2004 표준과 DO-178B 표준에서는 각각 위험성 분석과 시스템 안전 평가를 통하여 소프트웨어 무결성 등급과 소프트웨어 등급을 정하고 그에 따라 검증 활동의 강도를 정의하고 있다. IEEE 1012-2004 표준에서는 오류가 시스템에 미치는 영향력과 오류가 발생할 가능성에 따라 1 ~ 4 등급으로 분류하고 있다. 표 1, 표 2 는 분류 기준을 나타낸 것이다.. DO-178B 에서는 오류의 영향력에 따라 소프트웨어를 A ~ E 등급으로 분류하고 있다. 표 3 은 그 내용을 간략히 나타낸 것이다.

**표 1 IEEE 1012-2004 Software Integrity Level**

| Error | 오류 상태 발생 가능성 | | | |
|---|---|---|---|---|
| Consequence | Reasonable | Probable | Occasional | Infrequent |
| Catastrophic | 4 | 4 | 4 or 3 | 3 |
| Critical | 4 | 4 or 3 | 3 | 2 or 1 |
| Marginal | 3 | 3 or 2 | 2 or 1 | 1 |
| Negligible | 2 | 2 or 1 | 1 | 1 |

**표 2 IEEE 1012-2004 Definition of Consequences**

| Consequence | 정의 |
|---|---|
| Catastrophic | 인명 사망 위험, 완전한 임무 실패, 시스템의 안전성 손실, 대규모의 사회 및 재산 피해 |
| Critical | 큰 부상, 부분적 임무 실패, 시스템 손상, 사회 및 재산 피해 |
| Marginal | 심한 부상이나 질병, 보조 임무 수행 저하, 작은 규모의 사회 및 재산 피해 |
| Negligible | 작은 부상이나 질병, 시스템 성능에 작은 영향, 시스템 운영자의 불편 |

**표 3 DO-178B Software Level**

| Failure Condition | Level | Example |
|---|---|---|
| Catastrophic | A | 항공기 추락 위험성 |
| Hazardous | B | 승객 사망 위험성 |
| Major | C | 승객 부상, 항공기기 통제 불가 위험성 |
| Minor | D | 통제 가능한 고장 |
| No effect | E | 영향 없음 |

IEEE 1012-2004 의 소프트웨어 무결성 등급과 DO-178B 의 소프트웨어 등급은 개념적으로 크게 다르지 않다. DO-178B 의 E 등급의 경우 인증과 실제적인 관련이 없으므로, 사실상 A~D 등급으로 정의된다. 양 표준에서 사용하는 용어와 등급 분류 기준이 매우 비슷하며, 양 표준 모두 승객의 안전과 시스템 손실 여부 등에 따라 등급이 고려된다. 본 검증 절차에서는 IEEE 1012-2004 표준에서 정의하고 있는 4 단계 등급 분류 구조를 적용한다. 위험성 분석을 통하여 소프트웨어 등급을 분류하고, 등급에 따라 검증 활동을 수행한다. 정적 코드 분석은 모든 소프트웨어 등급에서 수행한다. 시험 활동은 소프트웨어의 기능을 확인하기 위하여 모든 등급에서 수행하되, 등급에 따라 세부 활동이 달라질 수 있다. 시험 커버리지 기준은 DO-178B 에 정의된 내용에 맞추어 적용하며, 등급별 커버리지는 표 4 의 내용과 같다. 정형 검증 활동은 높은 안전성을 확보할 수 있지만 많은 시간과 인력 등의 비용이 요구되므로, 최상위의 소프트웨어 등급일 경우에만 수행한다.

**표 4 DO-178B 등급별 코드 커버리지 요구 기준**

| Level | Coverage to be satisfied |
|---|---|
| A | MC/DC Coverage<br>Decision/Condition Coverage<br>Statement Coverage |
| B | Decision/Condition Coverage<br>Statement Coverage |
| C | Statement Coverage |
| D | N/A |

### 4.2. 역공학을 통한 정보 보완 및 복구

4.2.1. 개 요

일반적으로 소프트웨어 역공학은 소프트웨어의 유지보수와 재개발, 재구조화 등을 위한 목적으로 이용되고 있다 [9]. 본 검증 절차에서는 소프트웨어에 대한 검증을 진행하기 위해 소프트웨어의 부족한 정보를 보완하기 위한 방법으로 역공학을 적용하는 방안을 제시한다.

개발이 완료된 소프트웨어의 경우 개발 생명 주기에 맞추어 검증 절차를 진행하기가 어렵다. 개발 생명 주기의 앞부분에 해당하는 요구사항과 설계에 관한 정보가 완전하지 않을 수 있기 때문이다. 뿐만 아니라 요구사항과 설계에 관한 완전한 정보가 존재한다 하더라도 표준에서 요구하는 수준과 차이가 있을 수 있다. 따라서 검증 활동을 수행하기 전에 요구사항과 설계에 관한 정보들을 보완하고 복구하는 작업이 필요하다. 이를 위해 소프트웨어 역공학을 통해 요구사항을 분석하고, 프로그램의 구조를 분석하여 상세한 설계를 추출하고, 고수준의 설계 정보를 만들어낸다. 이렇게 복구된 요구사항과 설계 정보를 가지고 검토 과정을 거쳐 표준에서 요구하는 수준의 산출물을 작성한다. 그림 2 는 소프트웨어 역공학의 절차를 개념적으로 나타낸 것이다.

소프트웨어 역공학 과정은 반드시 거쳐야 하는 절차 단계는 아니다. 소프트웨어에 대한 정보가 이미 존재할 경우 부족한 부분이 있거나 표준의 요구에 충분히 부합하는지를 분석하고 이를 만족하는지를 먼저 확인해 보아야 한다. 분석 결과가 만족할 만한 수준이라면, 소프트웨어 역공학 과정을 거치지 않고 다음 단계로 진행할 수 있다.

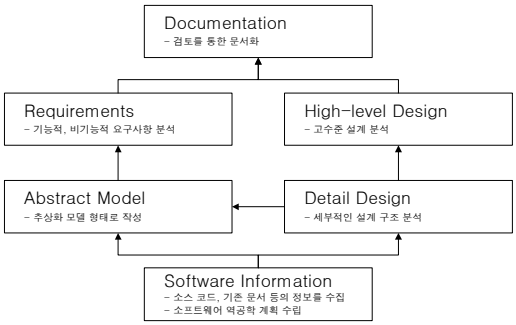


**그림 2 소프트웨어 역공학 절차**

4.2.2. 절 차

1) 설계 복구(Design recovery)

설계 복구는 소프트웨어의 기능과 구조, 고수준의 추상화 구조 정보를 재생성하는 작업이다. 이렇게 얻어진 설계 정보를 이용하여 검증을 계획하고 수행할 수 있게 된다. 설계 복구는 소프트웨어 정보 수집 및 분석, 구조 분석, 기능 분석, 고수준 구조 분석 단계로 이루어진다. 표 5 는 각 단계의 산출물과 활동에 대한 대략적인 설명이다.

**표 5 설계 복구 활동 및 산출물**

| 단계 | 입력 | 출력 | 활동 |
|---|---|---|---|
| 정보 수집 및 분석 | 소프트웨어 정보(문서, 소스 코드 등) | 설계 복구 계획 | 자료 수집 및 분석 설계 복구 계획 수립 |
| 구조 분석 | 소프트웨어 정보(문서, 소스 코드 등) | 구조 정보 (구조 도표 등) | 소프트웨어 구조를 분석하여 이해하기 쉬운 형태로 표현 |
| 기능 분석 | 구조 정보 | 기능 정보 | 구조 정보에 기능 정보를 분석하여 추가 |
| 고수준 구조 분석 | 구조 정보 | 고수준 구조 정보 (DFD, CFD 등) | 분석한 구조 정보를 바탕으로 고수준의 구조 정보를 작성 |

2) 요구사항 분석(Requirement analysis)

요구사항 분석은 소프트웨어를 분석하여 기능적, 비기능적 요구사항을 정의하는 작업이다. 소스 코드와 기존 문서 등 소프트웨어 정보와, 설계 복구를 통해 재생성된 정보를 바탕으로 요구사항 정보를 얻어내어, 검증을 진행하기 위한 정보로 이용한다. 요구사항 분석은 소프트웨어 정보 수집 및 분석, 추상화 모델 분석, 기능적 요구사항과 비기능적 요구사항 분석 단계로 이루어진다. 표 6 은 각 단계의 활동에 대한 대략적인 설명이다.

3) 문서화(Documentation)

복구된 요구사항과 설계 정보를 검토 하여 문서화 한다. 소프트웨어 요구사항 명세서(SRS: Software Requirement Specification), 소프트웨어 설계 기술서(SDD: Software Design Description), 소프트웨어 설계

**표 6 요구사항 분석 활동 및 산출물**

| 단계 | 입력 | 출력 | 활동 |
|---|---|---|---|
| 정보 수집 및 분석 | 소프트웨어 정보(문서, 소스 코드 등) | 요구사항 분석 계획 | 자료 수집 및 분석 요구사항 분석 계획 수립 |
| 추상화 모델 분석 | 구조 정보 | 추상화 모델 | 추상화 모델 작성 |
| 비기능적 요구사항 분석 | 구조 정보 | 비기능적 요구사항 | 구조 정보로부터 비기능적 요구사항 분석 |
| 기능적 요구사항 분석 | 추상화 모델 | 기능적 요구사항 | 모델로부터 기능적 요구사항 분석 |

명세서(SDS: Software Design Specification)등이 산출문서에 해당한다.

## 4.3. 정형 검증

4.3.1. 개 요

정형검증 기법은 하드웨어나 소프트웨어의 정확성을 증명하기 위해 사용되는 기법으로써, 특히 원자력이나 철도, 항공과 같이 안전필수 시스템의 안전성을 증명하기 위해 사용된다. OFP 는 항공 분야의 소프트웨어로, 그만큼 높은 소프트웨어 등급이 요구될 수 있다. 소프트웨어 등급이 높을 경우 그만큼 높은 안전성이 요구되므로 이를 검증하기 위한 절차에 정형검증 기법을 이용하여 요구사항에 맞는 설계 및 구현이 이루어 졌는지를 검증한다.

본 검증 절차는 자동화된 도구를 이용한 검증 기법을 이용한 검증절차이다. 정형 검증을 위한 기법으로 Model checking 기법 [10], [11]을 이용하여 검증을 수행한다. 그림 3 은 모델 체킹을 수행하기 위한 기본 절차를 나타낸 그림이다.

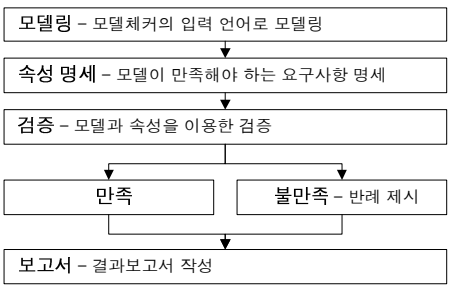


**그림 3 정형 검증 절차**

4.3.2. 모델링

시스템의 정확성을 검증하기 위한 첫 단계는 시스템이 가져야 하는 속성 명세이다. 예를 들면 'OFP 는 Deadlock 상황이 발생하면 안 된다'와 같은 속성이다. 위와 같은 형태의 중요한 속성들을 명세 했다면, 시스템의 정형 모델을 만드는 작업을 수행한다. 이를 위해 모델을 검증할 수 있는 모델 체커를 선정하고

**표 7 모델 체킹 도구**

| 도구 | 특징 | 입력 언어 | 속성 명세 |
|---|---|---|---|
| SPIN | 분산 소프트웨어의 검증에 적합 | PROMELA | LTL |
| UPPAAL | 실시간 시스템의 모델링 및 증명과 검증 수행에 적합 | Timed Automata | Query (CTL 과 유사) |
| VIS | 유한 상태 시스템의 시뮬레이션 및 정형 검증, 설계 | Verilog | CTL |
| NuSMV | 동기적, 비동기적 유한 상태 시스템의 표현 가능 | NuSMV input Language | CTL, LTL |
| BLAST | C 프로그램을 위한 모델 체킹 도구 | Program source | Temporal safety Property |

모델 체커의 입력 언어로 모델을 작성한다. 표 7 은 대표적으로 사용되는 모델 체커의 목록과 그 쓰임에 대해 나타내고 있다. 정형 모델은 검증하고자 하는 속성을 반영해야 하지만, 해당 속성의 정확성을 검증하는 작업에 영향을 주지 않는 속성들에 대해서는 추상화 작업이 이루어져야 한다.

4.3.3. 속성 명세

모델링 과정을 통하여 만들어진 모델을 검증하기 위하여 시스템이 만족해야 하는 요구사항을 명세해야 한다. 일반적인 모델 체킹 기법에서는 검증 속성 명세를 위해 Temporal Logic 을 이용한다. Temporal Logic 은 상태와 상태를 유발시키는 원인을 표현하도록 맞춰진 논리적 형태의 수식이다. 이 수식은 시간의 순서를 나타내는 기호를 포함하고 있다. 모델 체커에서 사용되는 Temporal Logic 의 대표적인 종류는 CTL*, CTL, LTL 이 있고, 이 중 CTL 과 LTL 이 많이 사용되고 있다.

4.3.4. 검증

모델 체커는 작성된 모델과 명세된 속성을 입력받아 해당 모델을 검증한다. 검증의 결과는 두 가지로 나뉜다. 하나는 모델이 검증하고자 하는 속성을 만족하는(Satisfied) 경우이고, 다른 하나는 만족하지 않는(Not Satisfied) 경우이다. 검증의 결과가 Not Satisfied 일 경우 모델 체커는 Counter-example 을 제시해 줌으로써 모델이 어떠한 상황에서 명세된 속성을 만족하지 않는지를 보여준다. 모델 체킹을 수행하는 전문가는 모델에서 오류가 나는 상태가 실제 시스템에서 어떻게 유발될 수 있는지를 명시해 주어야 한다.

4.3.5. 보고서

보고서는 사용된 모델 체커를 비롯하여, 모델링 과정, 추상화된 내용 및 방법, 검증 속성이 가지는 의미, 검증 과정 및 결과까지 모든 내용이 기록되어야 한다. 또한 검증 불만족인 검증 결과에 대한 분석 및 개선 방향을 제시할 수 있어야 한다.

## 4.4. 정적 코드 분석

4.4.1. 개 요

정적 코드 분석은 소프트웨어의 결점을 검출하기 위하여 소스 코드를 한 줄씩 검토하여 논리적 오류, 메모리 누수, 동적 할당, 코드상의 비효율성 등의 문제를 찾아내는 정적인 분석 방법이다. 정적 코드 분석을 수행함으로써 시험 활동에서 드러나지 않는 메모리 누수 위험, 변수 선언 및 초기화, 범위를 벗어난 데이터 등과 같은 문제들을 확인할 수 있다. 논리적인 오류와 비효율성, 메모리 누수 등의 문제는 내장형 소프트웨어에 있어서 민감한 부분이므로 정적 코드 분석을 통하여 더 높은 안전성을 확보할 수 있다.

4.4.2. 소프트웨어 검사(Inspection)

정적 코드 분석은 소프트웨어 검사(Inspection) 절차를 통하여 이루어진다. 검사 절차는 계획, 개요, 준비, 회의 과정과, 검출된 결점을 수정하는 재작업 단계와, 재검사 수행 여부를 결정하는 후속 작업(Follow-up) 단계로 이루어진다. 본 절차에서는 검증 결과를 개발에 반영할 수 없으므로 후속 작업 단계를 제외한다. 그림 4 는 검사 절차를 나타낸 것이다.

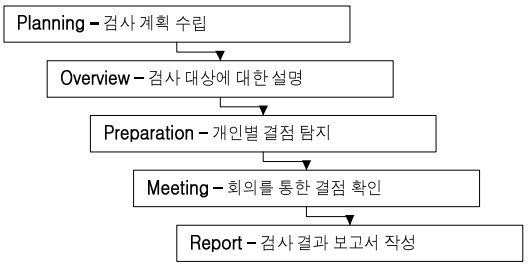


**그림 4 검사 절차**

검사를 명확하게 수행하기 위해서는 검사할 항목에 대한 목록(Checklist)을 작성하여 사용하여야 한다. 검사 목록은 분석을 수행할 때 어떤 부분에 초점을 맞추어 진행할지에 대한 기준을 제시한다. 검사 목록은 소프트웨어의 분야와 사용 프로그램 언어 등에 따라 달라질 수 있으며, OFP 의 특징과 개발 환경을 고려한 검사 목록이 작성되어야 한다. 표 8 은 검사 목록에 대한 대표적인 예이다. 정적 코드 분석은 자동화가 가능하며, 미리 작성한 검사 목록에 따라 검사를 수행한다. 여러 자동화 도구가 개발되어 있으므로 이를 이용하여 분석에 소요되는 시간을 줄일 수 있다.

**표 8 검사 목록 예시**

| Fault | 검사 목록 |
|---|---|
| Data faults | 변수 선언 여부를 확인한다.<br>변수 초기화 여부를 확인한다.<br>상수에 정의된 이름을 확인한다.<br>배열 경계 범위를 확인한다.<br>버퍼 오버플로우 가능성을 확인한다.<br>… |
| Control faults | 제어문 조건이 정확한지 확인한다.<br>반복문이 확실히 종료되는지 확인한다.<br>복합문의 괄호가 적절한지 확인한다.<br>… |
| Input/Output faults | 모든 입력 변수 사용 여부를 확인한다.<br>출력 전 변수의 값 지정 여부를 확인한다.<br>… |
| Interface faults | 함수와 메소드 호출의 매개변수의 개수와 순서가 올바른지 확인한다.<br>공유 메모리 구조를 확인한다.<br>… |
| Storage management faults | 동적 기억 장소 할당이 정확한지 확인한다.<br>기억 장소 해제가 명시적인지 확인한다.<br>… |
| Exception handling faults | 오류 조건이 정확한지 확인한다.<br>가능한 오류 조건들이 모두 고려되었는지 확인한다.<br>… |

## 4.5. 시 험

4.5.1. 개 요

시험은 대표적으로 사용되는 검증 활동으로써 소프트웨어를 동작시켜 소프트웨어의 기능과 정확성을 보장하는 방법이다. 요구사항 명세서와 설계 명세서에 작성된 내용을 모두 만족하는지를 확인하여, 소프트웨어가 요구사항을 만족하는 기능을 수행하고 동작에 이상이나 결함이 없음을 확인하는 활동이다. IEEE 1012-2004 표준과 DO-178B 지침 모두 주 검증 활동으로 시험을 제시하고 있으며 시험 세부 활동과 절차에 대한 내용을 정의하고 있다. OFP 를 위한 검증 절차에도 기능과 성능에 대한 검증을 위해 시험 활동을 구성하였다.

4.5.2. 시험 활동

IEEE 1012-2004 표준 에서는 4 수준 시험(컴포넌트 시험, 통합 시험, 시스템 시험, 합격 시험)으로 정의하고 있으며 소프트웨어 무결성 등급에 따라 각 수준 시험의 활동들을 정의하고 있다. DO-178B 에서는 단위 시험, 통합 시험, 블랙 박스 시험 및 합격 시험으로 정의하고 있으며 소프트웨어 등급에 따라 시험의 커버리지 적용 기준을 달리 정의하고 있다. 본 절차에서는 컴포넌트 시험, 통합 시험, 시스템 시험, 합격 시험의 4 수준 시험을 적용하며, 소프트웨어 등급에 따라 각 수준 시험의 기준을 달리 적용한다.

1) 컴포넌트 시험(Component test)

컴포넌트 시험은 소프트웨어 최소 단위를 대상으로 하는 시험으로 소프트웨어의 모듈의 기능이 정상적으로 동작하는지를 검증한다. 컴포넌트 시험에서는 기능 시험(Functional test)과 구조 시험(Structural test)을 수행할 수 있다. 기능 시험은 요구사항 명세에 따라 모듈의 입력 값과 출력 값의 집합을 정의하고, 입력 값에 따라 예상하는 대로의 출력 값이 나오는지를 확인하는 시험이다. 구조 시험은 모듈을 CFG 의 형태로 바꾸어 분기문의 조건, 경로 등을 확인하는 시험이다. 컴포넌트 시험은 소프트웨어 개발 환경에서 수행하며 따로 시험 환경을 구성하지 않는다.

2) 통합 시험(Integration test)

통합 시험은 소프트웨어의 각 모듈들 간의 통합을 확인하기 위한 시험으로 각 모듈간의 호출과 호출 시점, 인터페이스 사용, 데이터 전달 등이 제대로 이루어지는지를 검증한다. 모듈간의 상호작용은 굉장히 복잡하게 이루어져 있으며 하나의 모듈이 다른 모듈들의 상호작용에 새로운 문제를 가져올 수 있으므로, 통합 시험은 각 모듈의 통합 순서를 결정하여 하나의 모듈을 통합할 때 마다 반복적으로 수행하는 형태로 진행한다. 통합 시험은 컴포넌트 시험과 마찬가지로 소프트웨어 개발 환경에서 수행하며 따로 시험 환경을 구성하지 않는다.

3) 시스템 시험(System test)

시스템 시험은 소프트웨어가 요구사항을 만족하는지 확인하는 시험으로 기능 시험을 수행한다. 요구사항을 만족하는지를 검증해야 하므로 시스템 시험에서는 하드웨어의 환경이 중점적으로 고려되어야 한다. 따라서 하드웨어 중점적인 테스트 환경(HILS: Hardware In-the-Loop System)을 구성하여 시험한다. 기존 HELISCOPE Project 의 OFP 의 검증을 위해 FlightGear Simulator 를 적용하여 구성한 HILS 환경[4]에서 시스템 시험을 수행한다.

4) 합격 시험(Acceptance test)

합격 시험은 소프트웨어가 요구사항을 만족하는지를 실제 환경에서 확인하는 시험이다. 실제 환경에서의 소프트웨어 성능을 검증할 수 있으며, 소프트웨어와 하드웨어간의 상호작용과 인터페이스의 오류를 검증할 수 있다. 실제 환경에서 시험을 수행하여야 하므로, 시험을 수행하고 데이터를 원활하게 확인할 수 있는 사전 작업이 이루어져야 한다.

4.5.3. 절 차

시험 활동은 4 수준 시험을 검증 절차의 각 단계에 따라 진행한다. 각 수준의 시험은 시험 계획, 시험 설계, Test Case 작성, 시험 절차 작성, 시험 실행 활

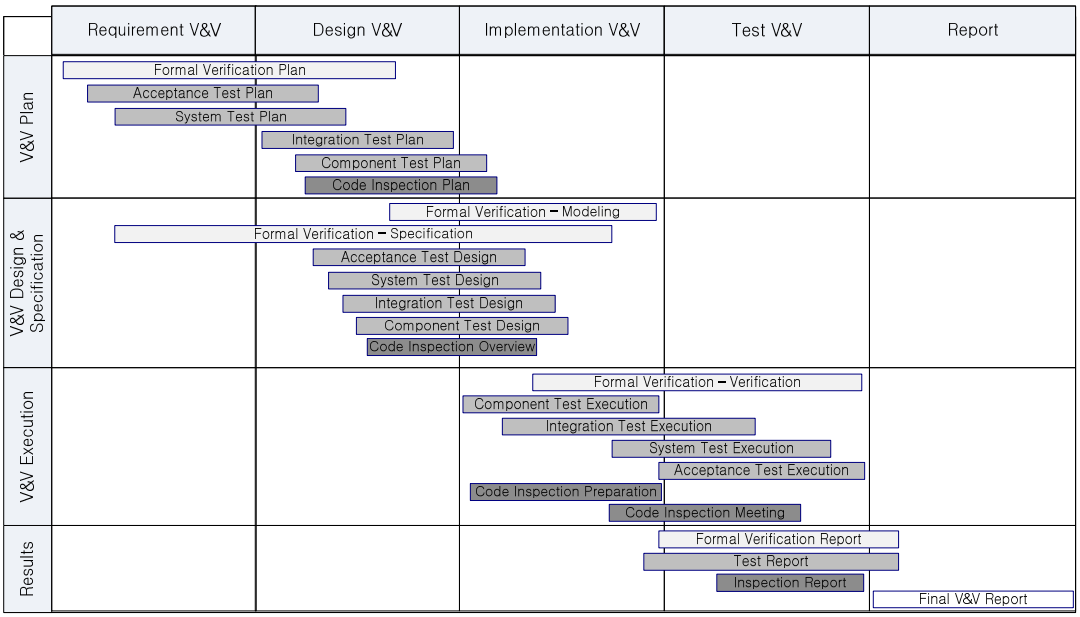


**그림 5 개발 생명 주기 별 검증 활동**

동으로 이루어진다. 검증 절차에서 시험의 세부 활동이 이루어지는 시점은 그림 5 에서 확인할 수 있다.

1) 요구사항 검증

요구사항 검증 단계에서는 합격 시험과 시스템 시험의 계획을 진행한다. 시험 계획을 통하여 시험의 목적과 시험할 아이템, 시험 수행을 위한 접근 방법, 시험 일정, 자원, 환경 등을 정의한다. OFP 의 시스템 시험을 위한 HILS 환경과, 합격 시험을 위한 실제 환경 구성 및 시험 접근 방법을 고려해야 한다.

2) 설계 검증

설계 검증 단계에서는 통합 시험과 컴포넌트 시험의 계획을 진행한다. 그리고 모든 수준의 시험의 설계를 시작한다. 시험 설계에서는 시험할 아이템의 특성을 정의하고 통과/실패 기준을 정의하며, 시험을 수행할 기법을 정의한다. 시험 수행 기법에는 OFP 의 특성을 고려하여 내장형, 실시간 소프트웨어를 위한 시험 기법이 사용되어야 한다.

3) 구현 검증

구현 검증 단계에서는 각 수준의 시험의 Test Case 를 작성한다. 입력 값과 출력 값의 집합을 정의하고 Test Case 를 작성하기 위한 기법을 선택한다. 통합 시험의 경우 입력 값 간의 관계나 시점 등의 내용도 정의될 수 있다. Test Case 를 수행하기 위한 환경적인 필요사항들도 정의한다. 동시에 시스템 시험, 통합 시험과 컴포넌트 시험 수행을 위한 절차를 정의하며, 시험 절차가 정의되면 컴포넌트 시험과 통합 시험을 수행한다.

4) 시험

'시험' 단계에서는 통합 시험 수행을 완료하고, 시스템 시험과 합격 시험을 위한 절차를 정의하고 이를 수행한다. 시험 단계에서는 모든 수준의 시험을 완료하고 시험 결과에 대한 문서를 작성한다.

4.5.4. 시험 문서

소프트웨어 시험관련 문서는 IEEE 829-1998 소프트웨어 시험 문서 표준 [12]에 따라 작성한다. 표준에서는 시험 계획서, 시험 설계 명세서, Test Case 명세서, 시험 절차 명세서, 시험상황기록서(Test log), 시험 사고 보고서, 시험 요약 보고서를 작성하는 것으로 정의하고 있다. 표준에 따라 각 시험 문서를 작성하여 시험의 계획과 진행 상황, 결과를 기록한다.

## 5. 결 론

항공 분야는 결함이 곧 치명적인 피해로 이어질 수 있는 안전성에 매우 민감한 분야이다. 따라서 항공기에 탑재되는 소프트웨어는 높은 안전성을 확보하여야만 하며, 이를 위해 엄격한 검증을 수행할 수 있는 검증 절차의 필요성이 요구된다. 국제적인 검증 표준의 내용을 참조하여 정형화된 검증 절차를 작성함으로써 소프트웨어의 높은 안전성을 확보할 수 있다. 본 논문에서는 항공기에 탑재되는 OFP 를 검증하기 위하여 개발 단계의 요구사항과 설계 정보를

보완하기 위한 소프트웨어 역공학 과정과, 안전성을 확보하기 위하여 정형 검증, 정적 코드 분석, 시험 활동으로 구성된 정형화된 검증 절차를 제시하였다. 이러한 검증 절차에 따라 검증을 수행함으로써 검증 대상에 대한 높은 안전성을 확보할 수 있을 것으로 기대한다. 향후 본 논문에서 제시한 검증 절차를 적용하여 HELISCOPE Project 의 OFP 를 대상으로 검증을 수행함으로써 OFP 의 결점을 탐지하여 높은 안전성을 확보한다.

## 참고문헌


[1] Doo-Hyun Kim, Kodirov Nodir, Chun-Hyon Chang, Jung-Guk Kim, "HELISCOPE Project: Research Goal and Survey on Related Technologies", International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pp.112~118, 2009.

[2] Se-Gi Kim, Seung-Hwa Song, Chun-Hyon Chang, Doo-Hyun Kim, Shin Heu, Jung-Guk Kim, "Design and Implementation of an Operational Flight Program for an Unmanned Helicopter FCC Based on the TMO Scheme", International Federation for Information Processing, pp.1~11, 2009.

[3] IEEE 1012-2004 Standard for Software Verification and Validation. 2005.

[4] SW Considerations in Airborne Systems and Equipment Certification RTCA/DO-178B. 1994.

[5] 김종섭, "비행제어시스템 설계 및 검증 절차", 제어 로봇 시스템학회 논문지 제 14 권, 제 8 호, pp.824~836, 2008.

[6] 이종복, 서상문, 금종용, 구인수, "고품질의 소프트웨어 개발을 위한 시험 지침", 한국산업경영시스템학회 2004 춘계학술대회 논문집, 2004.

[7] S.W.Cheon, J.S.Lee, K.C.Kwon, D.H.Kim, H.Kim, "The Software Verification and Validation Process for a PLC-based Engineered Safety Features-Component Control System in Nuclear Power Plants", 30th Annual Conference of IEEE Industrial Electronics Society, Vol.1, pp.827~831, 2004.

[8] Kim, K.H., Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", 18th IEEE Computer Software & Applications Conference, pp.392-402, 1994.

[9] Chikofsky, Elliot J., James H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software, Vol.7, Issue.1, pp.13~17, 1990.

[10] Edmund M. Clarke, Orna Grumberg and Doron A. Peled, "Model Checking", MIT Press, 2000.

[11] Béatrice Bérard, "Systems and software verification: model-checking techniques and tools", Springer, 2001.

[12] IEEE 829-1998 Standard for Software Documentation. 1998.