차 례

로봇 소프트웨어 시험평가 학습모듈의 개요

학습모듈의 목표

로봇 소프트웨어를 다양한 경우의 수에 대한 확인을 하여 잠재적으로 가지고 있을 수 있는 결함을 식별할 수 있다.

선수학습

프로그래밍 언어, 로봇공학 개론, 전기전자공학 개론, 메카트로닉스

학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 단위기능 시험평가하기	1-1. 단위기능 시험평가	1903080310_14v1.1	단위기능 시험평가하기
2. 통합기능 시험평가하기	2-1. 통합기능 시험평가	1903080310_14v1.2	통합기능 시험평가하기
3. 신뢰성 시험평가하기	3-1. 신뢰성 시험평가	1903080310_14v1.3	신뢰성 시험평가하기

핵심 용어

소프트웨어 테스트, 단위 시험, 통합 시험, 신뢰성 시험, 정적 분석, 동적 분석, 테스트 케이스, 자동화 도구

학습 1 단위기능 시험평가하기

학습 2 통합기능 시험평가하기
학습 3 신뢰성기능 시험평가하기

1-1. 단위기능 시험평가

학습 목표

- 단위기능의 정상적 수행 여부를 판단할 수 있는 시험 절차서를 작성할 수 있다.
- 시험 절차서에 따라 단위기능을 구현한 함수의 동작이 적절한지 판단할 수 있다.
- 단위기능을 시험한 결과를 보고서로 작성할 수 있어야 한다.

필요 지식 /

① 프로그램 언어

프로그램 언어는 컴퓨터 시스템을 동작시키는 프로그램을 작성하기 위한 언어이며, 사람이 사용하기 편리한 정도에 따라 저급언어와 고급언어로 분류된다.

<표 2-1> 프로그램 언어의 종류

구분	프로그램 언어 예	특징
저급언어	기계어, 어셈블리어	기계 중심적인 언어 컴퓨터 시스템마다 달라 호환이 어려움.
고급언어	C, C++, ADA, JAVA 등	인간이 사용하는 자연어와 닮음. 대부분의 언어가 고급언어임.

이 중 로봇 개발 시 주로 많이 사용되는 프로그램 언어는 C, C#, C++, JAVA, Python 등이다.

1. C 언어

C는 1972년 켄 톰슨과 데니스 리치가 벨 연구소에서 일할 당시 새로 개발된 유닉스 운영체제에서 사용하기 위해 개발한 프로그래밍 언어이다. 오늘 날 많은 운영체제의 커널이 C 언어로 작성되었고, C 표준 라이브러리가 국제표준화기구에 의해 채택되었으며, C++, C# 등의 확장형 언어의 뿌리가 되었다.

2. JAVA

자바는 썬 마이크로시스템즈의 제임스 고슬링(JAMES GOSLING)과 다른 연구원들이 개발한 객체 지향적 프로그래밍 언어이며, 무료로 제공되고 있다. 처음에는 가전제품 내에 탑재해 동작하는 프로그램을 위해 개발했지만 현재 웹 애플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용하고 있다.

자바의 개발자들은 유닉스 기반의 배경을 가지고 있었기 때문에 문법적인 특성은 파스칼이 아닌 C 언어와 비슷하다. 자바를 다른 컴파일 언어와 구분 짓는 가장 큰 특징은 컴파일된 코드가 플랫폼 독립적이라는 점이다. 자바 컴파일러는 자바 언어로 작성된 프로그램을 바이트코드라는 특수한 바이너리 형태로 변환한다. 바이트코드를 실행하기 위해서는 JVM(자바 가상 머신, java virtual machine)이라는 특수한 가상 머신이 필요한데, 이 가상 머신은 자바 바이트코드를 어느 플랫폼에서나 동일한 형태로 실행시킨다. 때문에 자바로 개발된 프로그램은 CPU나 운영체제의 종류에 관계없이 JVM을 설치할 수 있는 시스템에서는 어디서나 실행할 수 있으며, 이 점이 웹 애플리케이션의 특성과 맞아 떨어져 폭발적인 인기를 끌게 되었다.

3. PYTHON

파이썬(PYTHON)은 1991년 프로그래머인 귀도 반 로섬(GUIDO VAN ROSSUM)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체 지향적, 동적 타이핑(dynamically typed) 대화형 언어이다.

파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발모델을 가지고 있다. C언어로 구현된 C파이썬 구현이 사실상의 표준이다.

2 소프트웨어 시험 관련 용어

1. 소프트웨어 시험

IEEE STD 829 에서는 소프트웨어 시험의 정의를 아래와 같이 정의하고 있다.

"요구된 상태와 현재 개발된 상태 사이의 차이점(즉, 버그)을 발견하기 위해 소프트웨어를 분석하고 평가하는 프로세스(The process of analyzing a software item to detect the differences between exiting and required conditions(that is, bugs) and to evaluate the features of the software item)"

2. 확인(verification)과 검증(validation)

(1) 확인(verification)

확인은 한 마디로 하면 제품을 올바르게 만드는지?(Are we building the product right?)를 확인하는 과정이다. 제품이 설계 사양에 부합되게 제작되고 있는지를 확인한다.

(2) 검증(Validation)

검증은 올바른 제품을 만드는가?(Are we building the right product?)라는 관점에서 바라보며 설계 사양이 사용자의 요구 사항에 부합되는지를 검증한다.

3. 결함 관련

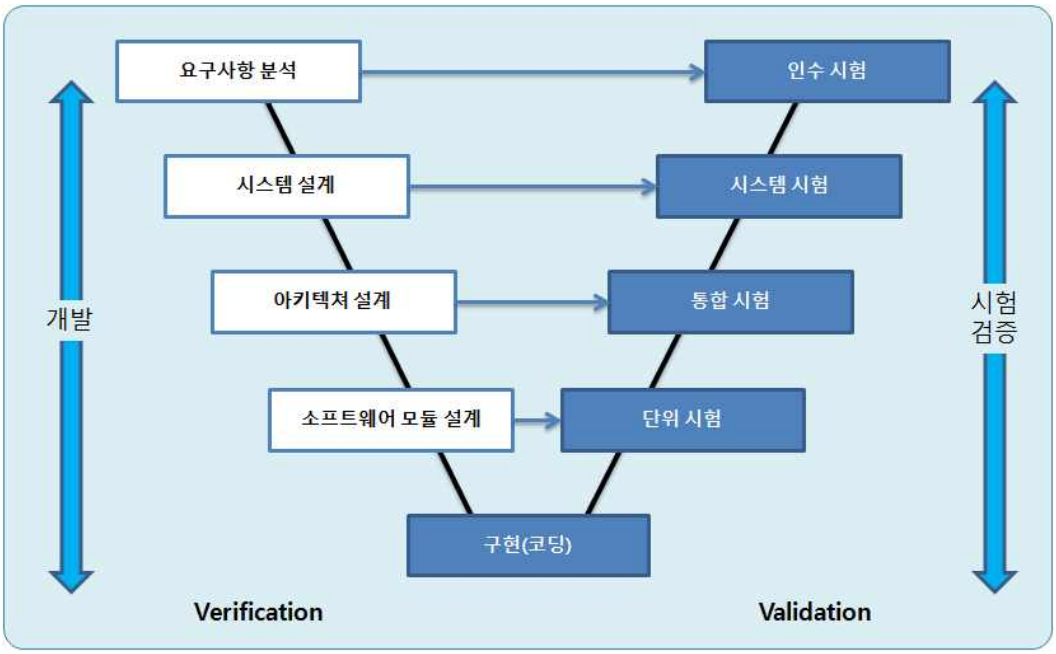
- 에러, 오류(error): 요구 사항을 잘못 이해하여 발생하는 실수
- 결함(fault): 에러가 원인이 되어 프로그램에 반영된 결과
- 실패(failure): 결함이 있는 부분이 실행될 때 발생되는 현상

3 소프트웨어 시험 종류

소프트웨어 시험에는 시험의 단계별 혹은 시험 목적이나 시험 방법 등에 따라 매우 다양하게 구분할 수 있다.

1. 시험 단계(test level)

소프트웨어 개발 수명 주기 V-모델의 단계에 따라 단위 시험, 통합 시험, 시스템 시험, 인수 시험 등으로 구분한다. 시험 검증은 개발의 역순으로 진행한다.



[그림 1-1] V-모델

<표 1-2> 단계별 시험의 종류

순번	구분	내용	검증
1	단위 시험	각 컴포넌트에 대한 시험으로 세부 구현 클래스와 함수에 대한 검증	알고리즘, 로직, 정확도 및 정밀도, 예외 처리 등
2	통합 시험	컴포넌트 간 기능 연동을 위한 검증	매개변수 전달, 전역데이터 손상, 동작 순서 오류 등
3	시스템 시험	기능 시험 및 비기능 시험을 포함한 배포 환경에 대한 검증	인터럽트 핸들링, 데이터버스/자원 연결 문제, 자체 고장 탐지 등
4	인수 시험	요구 사항 충족 확인	기능/성능/운용 요구 조건 등

2. 시험 방법 - 코드 수행 여부에 따른 구분

실제로 작동하는 런타임 소프트웨어로 시험을 하느냐 아니면 문서나 코드와 같이 정적인 자료를 가지고 검증을 하느냐에 따라 정적 시험과 동적 시험으로 분류할 수 있다.

(1) 정적 시험

프로그램 코드에 대한 정적 시험은 사람 중심의 코드리뷰 방법과 자동화 시스템에 의한 정적 분석 방법으로 크게 나뉜다.

<표 1-3> 정적 시험 방법

방법	특징
코드리뷰	개발 과정에서 가장 쉽게 적용할 수 있는 방법 리뷰어(참여하는 사람)의 능력이나 참여 정도에 따라 결과 편차 큼. 피어 리뷰, 패스 어라운드, 팀 리뷰, 워크스루, 코드 인스펙션 등
정적 분석	자동화된 도구에 의해 수행됨. 코드 복잡도 계산, 코딩 규칙 위배 및 사용하지 않는 데드코드 검출 결과의 편차가 거의 없음. 오픈소스 도구 및 상용화 버전 있음.

(2) 동적 시험

동적 시험은 코드를 실행하면서 수행하는 시험으로 코드 실행률(code coverage)이 주요 시험의 지표가 된다.

3. 시험 방법 - 박스 구분

(1) 블랙박스 시험

블랙박스 시험은 사용자의 시각으로 평가하는 방법으로 테스트팀 위주로 이루어지는 시험이며 모든 단계의 시험이 다 가능하나 일반적으로 시스템 시험과 인수 시험이 대상이 된다.

이 테스트는 내부 구조나 작동 원리를 모르는 상태에서 입력과 출력값만으로 확인하는 시험이다. 다만 해당 프로그램이 어떠한 기능을 가지고 있으며 어떻게 동작하는가를 정의한 요구 사항 명세서나 설계 명세서를 참조하여 테스트한다.

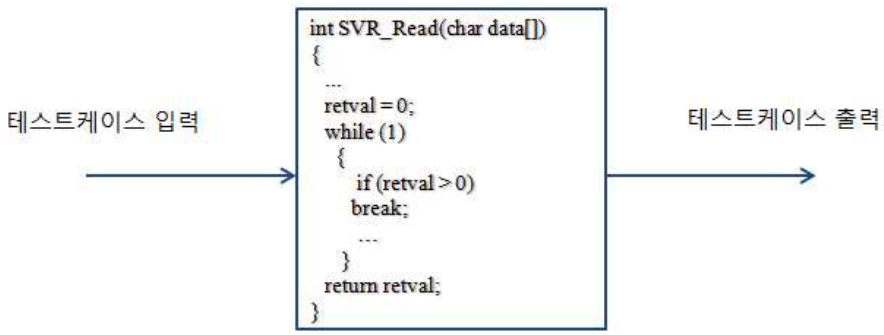
이 시험을 통하면 기능상의 문제나 인터페이스 문제 혹은 database와 관련된 에러를 발견할 수 있다. 그러나 프로그램의 내부 로직을 모르는 상태로 시험을 하기 때문에 결함의 요인을 발견하기는 화이트박스 시험보다 상대적으로 어렵다.

[그림 1-2] 블랙박스 시험 개념도

(2) 화이트박스 시험

화이트박스 시험은 시스템의 내부 구조를 이해하고, 내부 로직 구조를 확인하는 시험으로 프로그램 코드들의 논리적인 모든 경로를 검사하는 방법이다. 개발팀 주도로 이루어지는 시험이며 보통 단위 시험과 통합 시험이 대상이 된다.

화이트박스 시험은 프로그램을 실행시키지 않고도 프로그램 구현 논리나 조건에 관한 사항, 내부 자료 구조에 대한 것들을 시험할 수 있다.



[그림 1-3] 화이트박스 시험 개념도

4. 시험 유형

(1) 기능 시험(functional test)

소프트웨어가 가지고 있는 본연의 목적에 해당되는 기능에 대한 시험을 의미한다. 해당 소프트웨어의 기능은 요구 사항 명세서나 설계 명세서 등에 정의되어 있으며 모든 시험 단계에서 수행될 수 있다.

(2) 비기능 시험(non-functional test)

비기능 시험은 성능 시험, 장애 시험, 안정성 시험, 확장성 시험, 신뢰성 시험 등 소프트웨어가 가지고 있어야 하는 기능 외에 소프트웨어 제품의 특성에 대한 시험이라고 생각할 수 있다.

<표 1-4> 비기능 시험의 종류

시험 구분	정의
성능 시험	시스템에 부하를 주면서 요구하는 성능 지표를 확인하는 시험
장애 시험	시스템에 장애가 생겼을 때 이의 감지와 장애 복구 능력을 검증하는 시험
안정성 시험	시스템이 오랫동안 운영이 되더라도 문제가 없는지 검증하는 시험
확장성 시험	시스템을 증설함에 따라 용량이 선형적으로 증가하는지 확인하는 시험

(3) 확인 시험(confirmation test)

확인 시험은 다양한 시험 과정에서 발견된 결함을 해결한 후에 결함이 제대로 해결되었는지 검증하는 과정이다.

(4) 회귀 시험(regression test)

회귀 시험은 앞 단계에서 정상적으로 수행된 시험을 반복하는 과정이다. 이 시험은 이미 시험이 수행된 코드가 아닌 다른 부분의 결함 해소나 소프트웨어의 변경 혹은 추가 진행 등으로 기존에 잘 작동하던 기능이 영향을 받지는 않았는지 검증하는 과정이다.

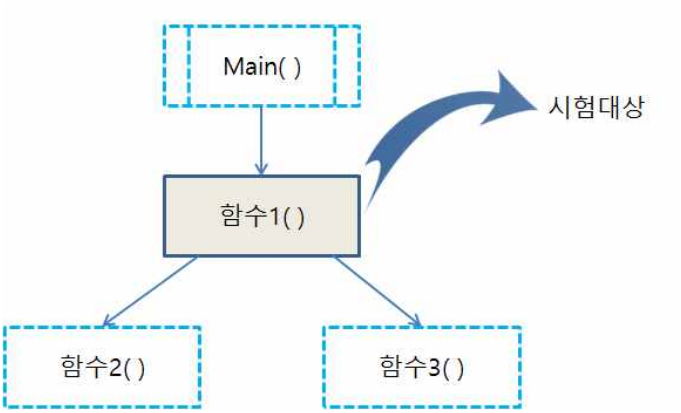
소프트웨어 시험에는 이 외에도 매우 다양한 종류의 시험들이 있다.

요즘과 같이 기능이 많고 복잡한 소프트웨어는 버그나 결함이 전혀 없는 상태로 만드는 것이 거의 불가능하다. 소프트웨어 시험은 작은 비용과 시간을 투입하여 최대한 효율적으로 최대한 많은 결함을 찾아내는 것을 목표로 해야 한다. 따라서 전체 시험을 계획할 때 무슨 시험을 언제 어떠한 방법으로 수행할지를 정하는 것이 매우 중요하다.

4 단위기능 시험

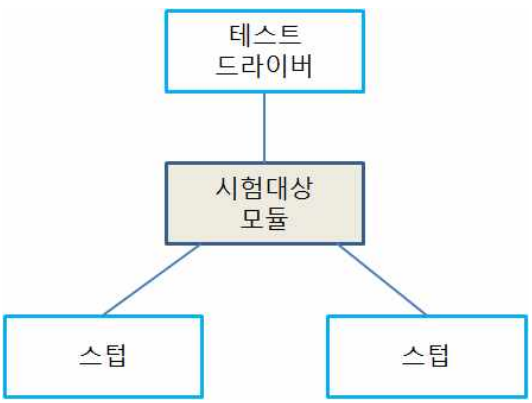
1. 단위 시험 정의

단위 시험은 프로그램을 구성하는 가장 작은 구성 요소(클래스나 함수 단위)를 대상으로 하며, 소스 코드의 각 함수가 프로그래머의 의도대로 정확히 작동하는지를 확인 검증하는 시험이다. 시험의 방법은 아래와 같이 시험 대상 함수를 다른 함수들로부터 분리시켜 시험을 한다.



[그림 1-4] 시험 대상 함수 분리

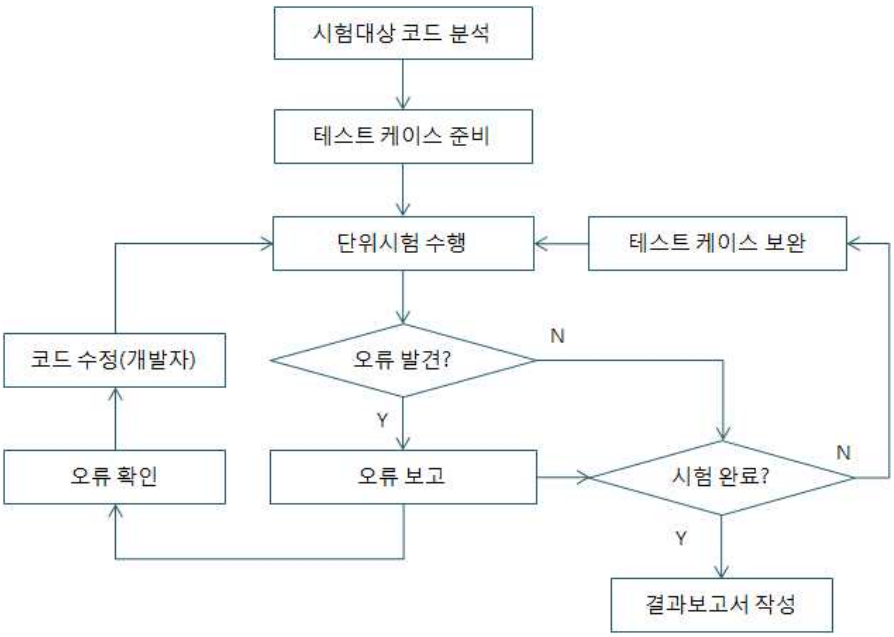
그러나 시험을 실시할 대상 함수가 바로 실행되지 않는 경우가 있기 때문에 테스트 드라이버를 만들어 시험 대상 모듈을 호출해야 할 경우도 있고, 하나의 모듈을 시험한다고 해도 하나의 모듈이 다른 모듈을 호출할 경우 시험 대상이 되는 모듈은 완전하게 실행되지 않을 수 있어 시험 대상 모듈이 호출하는 모듈인 스텁을 가상으로 만들어야 할 경우도 있다. 따라서 단위 시험을 수행하기 위해서는 아래와 같이 필요에 따라 테스트 드라이버와 스텁을 만들어 시험해야 한다. 테스트 케이스를 작성할 때 이에 대한 계획도 수립하여야 한다.



[그림 1-5] 스텁과 테스트 드라이버

2. 단위 시험 수행 절차

단위 시험의 수행 절차는 아래와 같다.



출처 : http://blog.naver.com/suresofttech/220694868279 2016. 07. 03. 검색
[그림 1-6] 단위 시험 절차

(1) 시험 환경 구축

소프트웨어 시험을 하기 위해서는 시험 환경을 구축하여야 하며 이를 위해서는 개발 환경을 알아야 한다. 하드웨어 아키텍처, 운영체제, 개발 프로그래밍 언어 및 통합개발 환경과 사용 컴파일러 등에 대해 조사하여 이에 맞게 시험 환경을 구축하여야 한다.

(2) 기능 및 코드 분석

해당 소프트웨어가 가지고 있는 기능과 코드가 어떤 방식으로 구현되어 있는지를 분석하여 테스트 케이스 작성 시 참고하여야 한다. 이를 위해 요구 사항 분석 자료와 설계 자료를 참조한다.

(3) 테스트 케이스 작성

실제 테스트를 수행할 시 시험 대상 함수에 입력값을 주고, 그 실제 출력값이 예상 출력값과 같은지를 비교하는 방식으로 시험을 진행한다. 그러기 위해서는 입력값과 예상되는 출력값을 미리 정하게 되는데 이것을 테스트 케이스라고 한다. 또 테스트 드라이버와 스텁에 대한 정의도 수행하여야 한다.

(4) 결과보고서 작성

모든 시험이 완료된 후에는 이를 종합 정리하여 결과보고서를 작성하여야 한다. 전체 시험 계획에 식별되어 있는 많은 시험 중 이 보고서를 근거로 단위 시험 완료를 확인한다.

5 단위 시험 절차서

1. 시험 계획서

소프트웨어 개발이 완료되고 시험 단계에 들어가기 전에 소프트웨어 시험에 대한 계획을 수립한다. IEEE STD 829에서는 마스트 테스트 플랜(master test plan)과 레벨 테스트 플랜(level test plan)을 규정하고 있다. 이 계획에는 시험 대상 소프트웨어 품목, 시험 환경, 수행될 시험 내용, 시험 방법과 범위가 정의되어 있다. 규모가 큰 개발 프로젝트의 경우는 시험 계획서가 먼저 나오고 이 계획을 바탕으로 시험 절차서를 작성한다.

<표 1-5> 시험 계획서 (사례)

ooo 장비
소프트웨어 시험 계획서
(Software Test Plan)

1. 개요
   - 장비 설명, 개발 일정 등
2. 시험 환경
   - 시험 대상 품목, 수행될 시험, 시험 조직 등
3. 시험 식별
   - 시험 수준, 시험 분류, 시험 조건, 시험 계획 등
4. 일정
   - 시험 일정 계획 및 장소
5. 요구 사항 확인 방법
   - 요구 사항 추적성, 식별자 정의

:

2. 시험 대상 프로그램 파악

시험을 수행하기 전에 처음으로 해야 하는 것은 시험 대상 프로그램을 파악하는 것이다. 만일, 개발자가 단위 시험을 수행한다면 이 과정이 필요 없겠지만 개발자가 아닌 제 3자가 시험을 수행한다면 반드시 필요한 과정이다. 대상 프로그램의 목적과 기능은 무엇인지? 어떤 하드웨어 플랫폼에서 동작하는지? 개발환경은 어떠했는지? 등에 대해 구체적으로 알아야 시험을 정확히 수행할 수 있다.

시험 대상 프로그램을 파악하는 가장 좋은 방법은 아래와 같은 개발과 설계 단계에서의 산출물을 확인하는 것이다.

- 운용 개념 기술서
- 개발 계획서
- 요구 사항 명세서
- 설계 기술서
  - 사용 사례 다이어그램(usecase diagram)
  - 클래스 다이어그램(class diagram)
  - 순차 다이어그램(sequence diagram) 등

이를 통해서, 아래와 같은 내용을 파악한다.

(1) 프로그램 사용 환경

<표 1-6> 프로그램 사용 환경 (사례)

구분	내용 (예)
프로그램 명	AAA ver 0.1
사용 체계	BBB 제어 컨트롤러 시리즈
적용 프로세서	DSP TMSXXXXXX
적용 운영체제/BIOS	DSP/BIOS
:	:

(2) 개발환경 및 코드 정보

<표 1-7> 프로그램 개발환경 및 코드 정보 (사례)

구분	내용 (예)
칩(CPU/MCU)	TMS320C6713
개발언어	C
개발환경	Code Composer Studio
컴파일러	c6x.exe
라인 수	00,000 line
파일 수	00개
함수 수	00개
:	:

(3) 시험 대상 항목 추출

시험 대상 프로그램의 함수 목록을 추출한다.

<표 1-8> 함수 목록 (사례)

모듈 명	번호	함수 명 (예)	설명(예)
RunMain	1	GetCmd	명령값을 가지고 온다.
	2	SaveData	입력된 데이터를 저장한다.
	3	doCommand	명령을 실행한다.
	4	SetSpeed	속도값을 설정한다.
	5	GetSpeed	속도값을 읽어 들인다.
	6	Emergency_Stop	긴급 중지한다.
	:	:	:
FlashCmd	21	InitFlash	Flash ROM을 초기화한다.
	22	BurnFlash	Flash ROM에 데이터를 굽는다.
	23	EraseFlash	Flash ROM을 지운다.
	:	:	:

3. 단위 시험 절차서 작성

시험 절차서에는 아래와 같은 사항이 포함되어야 하며, 적절히 테일러링해서 사용할 수 있다.

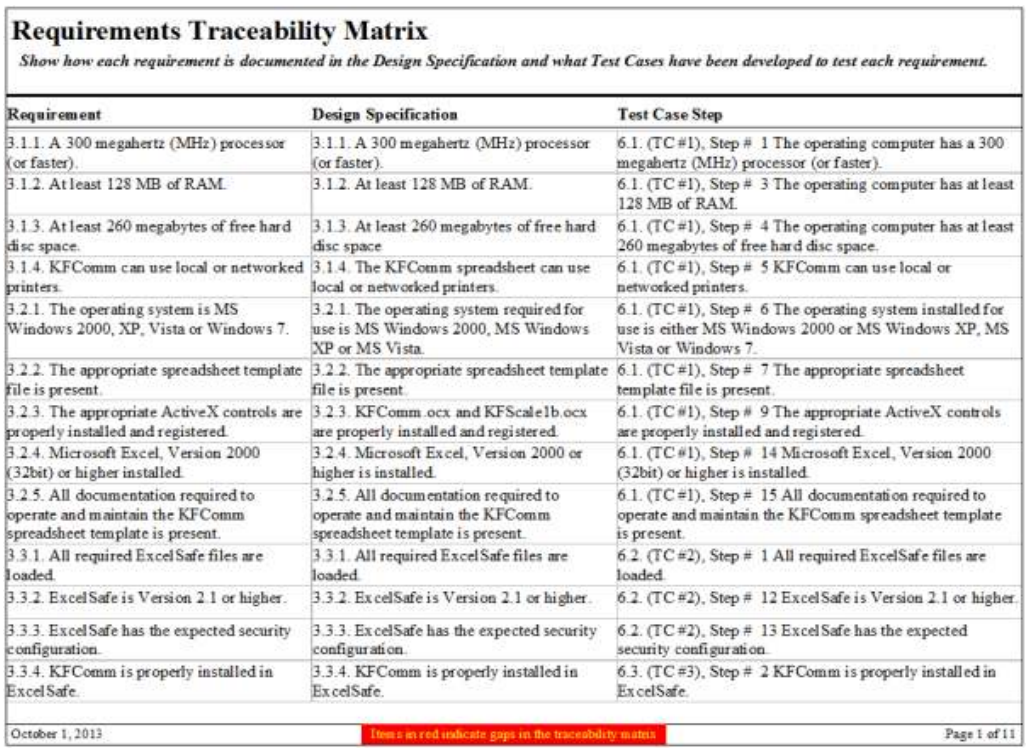
- 시험 개요
  - 시험 목적과 시험 항목
  - 시험 환경 및 필요 장비
- 개발환경
- 테스트 케이스
  - 시험 입력 자료
  - 예상 결과
  - 평가 기준
- 시험의 순서
- 요구 사항 추적성 매트릭스

(1) 요구 사항 추적성 매트릭스

설계 단계에서 최초 시스템의 요구 사항을 하위 단계에 맞는 요구 사항으로 도출해서 구현해야 하며 각 단계에서 시험 시에는 그 단계에 맞는 요구 사항을 확인하여 시험하여야 한다.

각 단계에서 필요한 요구 사항과 시험 항목을 매트릭스로 표현해 그 추적성을 확인하는 표를 요구 사항 추적성 매트릭스(requirements tracebility matrix)라고 한다. 하위 단계의 요구 사항 추적성 매트릭스는 상위의 요구 사항 추적성 매트릭스의 일부가 될 수 있다.

요구 사항 추적성 매트릭스는 다양한 형태로 작성될 수 있으나 요구 사항과 테스트 케이스는 반드시 들어가야 한다.



Requirements Traceability Matrix

Show how each requirement is documented in the Design Specification and what Test Cases have been developed to test each requirement.

Requirement	Design Specification	Test Case Step
3.1.1. A 300 megahertz (MHz) processor (or faster).	3.1.1. A 300 megahertz (MHz) processor (or faster).	6.1. (TC #1), Step # 1 The operating computer has a 300 megahertz (MHz) processor (or faster).
3.1.2. At least 128 MB of RAM.	3.1.2. At least 128 MB of RAM.	6.1. (TC #1), Step # 3 The operating computer has at least 128 MB of RAM.
3.1.3. At least 260 megabytes of free hard disc space.	3.1.3. At least 260 megabytes of free hard disc space	6.1. (TC #1), Step # 4 The operating computer has at least 260 megabytes of free hard disc space.
3.1.4. KFComm can use local or networked printers.	3.1.4. The KFComm spreadsheet can use local or networked printers.	6.1. (TC #1), Step # 5 KFComm can use local or networked printers.
3.2.1. The operating system is MS Windows 2000, XP, Vista or Windows 7.	3.2.1. The operating system required for use is MS Windows 2000, MS Windows XP or MS Vista.	6.1. (TC #1), Step # 6 The operating system installed for use is either MS Windows 2000 or MS Windows XP, MS Vista or Windows 7.
3.2.2. The appropriate spreadsheet template file is present.	3.2.2. The appropriate spreadsheet template file is present.	6.1. (TC #1), Step # 7 The appropriate spreadsheet template file is present.
3.2.3. The appropriate ActiveX controls are properly installed and registered.	3.2.3. KFComm.ocx and KFScale1b.ocx are properly installed and registered.	6.1. (TC #1), Step # 9 The appropriate ActiveX controls are properly installed and registered.
3.2.4. Microsoft Excel, Version 2000 (32bit) or higher installed.	3.2.4. Microsoft Excel, Version 2000 or higher is installed.	6.1. (TC #1), Step # 14 Microsoft Excel, Version 2000 (32bit) or higher is installed.
3.2.5. All documentation required to operate and maintain the KFComm spreadsheet template is present.	3.2.5. All documentation required to operate and maintain the KFComm spreadsheet template is present.	6.1. (TC #1), Step # 15 All documentation required to operate and maintain the KFComm spreadsheet template is present.
3.3.1. All required ExcelSafe files are loaded.	3.3.1. All required ExcelSafe files are loaded.	6.2. (TC #2), Step # 1 All required ExcelSafe files are loaded.
3.3.2. ExcelSafe is Version 2.1 or higher.	3.3.2. ExcelSafe is Version 2.1 or higher.	6.2. (TC #2), Step # 12 ExcelSafe is Version 2.1 or higher.
3.3.3. ExcelSafe has the expected security configuration.	3.3.3. ExcelSafe has the expected security configuration.	6.2. (TC #2), Step # 13 ExcelSafe has the expected security configuration.
3.3.4. KFComm is properly installed in ExcelSafe.	3.3.4. KFComm is properly installed in ExcelSafe.	6.3. (TC #3), Step # 2 KFComm is properly installed in ExcelSafe.

출처 : http://www.ofnisystems.com/products/fastval/details/traceability-matrix/ 2016. 08. 03. 스크린샷.
[그림1-7] 요구 사항 추적성 매트릭스 (예시)

(2) 테스트 케이스

단위 시험을 준비하는 절차 중 가장 많은 시간이 소요되고 시험 결과 도출에 가장 중요한 절차가 테스트 케이스 작성이다. 테스트 케이스를 작성하기 위해서는 코딩을 이해할 수 있는 지식과 대상 프로그램의 원리와 구조에 대해서도 어느 정도 이해가 있어야 한다.

(가) 테스트 케이스 작성 항목

테스트 케이스를 작성할 경우 보통 아래와 같은 항목들로 구성되나 필요 없는 항목은 삭제하거나 더 필요한 항목을 추가하여도 된다.

- Test Case ID
- Test Case 에 대한 상세 설명
- 테스트 절차 / 실행 순서
- 관련 요구 사항
- 작성자
- Pass / Fail
- 기타 특이 사항

(나) 좋은 테스트 케이스 작성법

좋은 테스트 케이스 작성법은 아래와 같다.

- 가능한 한 한 번에 1개의 기능을 시험하게 작성할 것
- 포지티브 시나리오와 네거티브 시나리오를 전부 커버할 수 있도록 작성할 것
- 추상적이지 않고 명확하고 쉽게 이해할 수 있도록 작성할 것
- 목적을 명확히 할 것
- 필요 없는 절차 작성하지 말도록 할 것
- 요구 사항 추적성을 유지하도록 작성할 것
- 반복 시험할 수 있도록 필요하다면 다시 사용 가능하게 작성할 것

⑥ 자동화 도구를 이용한 단위 시험

요즘처럼 소프트웨어가 기능이 많아지고 복잡해지면 사람이 모든 시험을 수행하고 관리하기가 쉽지 않아 자동화 도구를 사용하는 게 일반적이다. 자동화 도구에 대해서는 '학습 내용 3-2' 를 참조한다.

자동화 도구는 시험 수행과 관리를 편리하게 해주는 도구이지 시험에 대한 판단까지 해주는 만능 도구가 아니다. 자동화 도구는 프로그램 코딩 규칙이나 메모리 누수 같은 오류는 식별해 줄 수 있지만, 프로그램상의 로직까지는 판단해 주지 못한다. 이는 시험자가 확인해서 찾아내야 할 몫이다.

수행 내용 / 단위 기능 시험 수행하기

재료 · 자료

- 노트 및 필기도구
- 시험 절차서 샘플
- 시험 결과보고서 샘플

기기(장비 · 공구)

- 컴퓨터, 윈도우즈 XP 이상
- 사무용 프로그램 (MS Office)
- 프린터
- 인터넷

안전 · 유의 사항

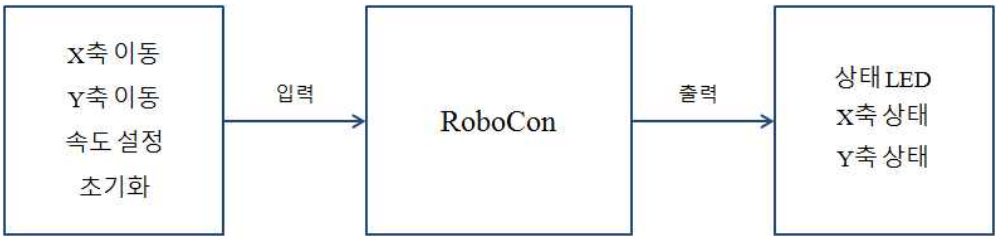
- 공개 소프트웨어를 다운로드하여 사용할 경우 라이선스를 반드시 확인하여 사용 여부를 결정하여야 한다. 일부 소프트웨어는 프리웨어였다가 셰어웨어로 혹은 유료 판매 정책으로 수정할 수도 있다.
- 학습을 위해 다운로드한 프로그램은 학습이 완료된 후 학습장의 지침을 확인한 후 필요시 프로그램을 제거해야 한다.

수행 순서

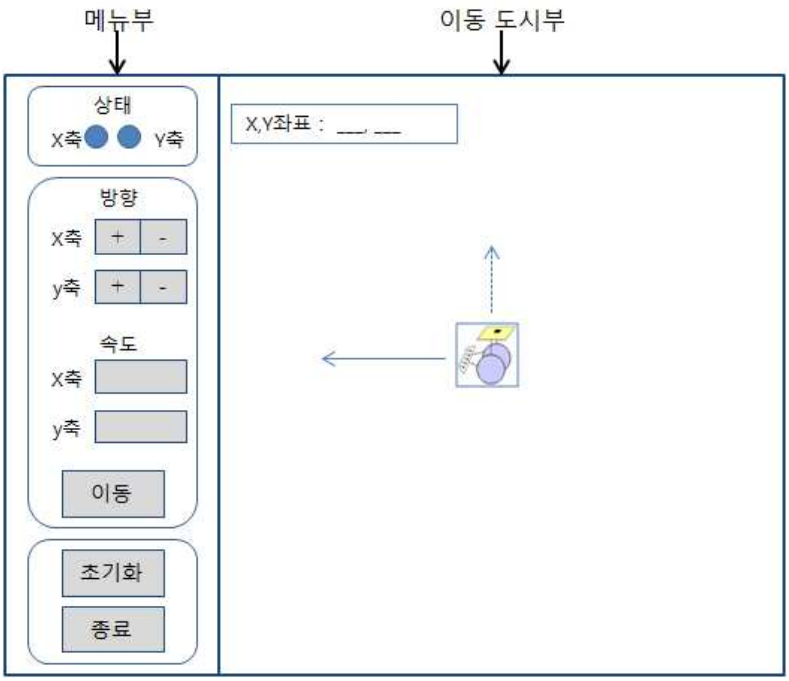
1 시험 대상 프로그램의 예제를 작성한다.

단위 시험을 수행하기 위해서는 시험 대상 프로그램이 반드시 있어야 한다. 간단한 기능을 가진 샘플 프로그램을 작성하고 이를 가지고 단위 시험을 수행한다.

1. 프로그램 개요
   - 프로그램명: 프로그램 이름은 프로그래머 임의로 명명한다. 여기서는 RoboCon으로 명명한다.
   - 운영 환경: 학습 시 제공된 컴퓨터 운영체제에서 구동될 수 있도록 한다.
   - 프로그램 언어: C 혹은 C++
   - 프로그램 개요: 로봇의 움직임을 묘사하는 프로그램
   - 프로그램 기능: 사용자로부터 명령을 입력받아 이동을 묘사하고 상태값을 출력한다.



[그림1-8] 작성 프로그램 블록도



[그림1-9] 작성 프로그램 화면 예시

위 프로그램의 화면은 예시이며, 각 프로그래머의 취향에 따라 위치, 표현 방법, 제어 방법 등을 다르게 구현한다.

2. 요구 사항

이 프로그램의 요구 조건은 아래와 같다.

- 사용자는 로봇의 현재 위치를(X, Y축) 알 수 있어야 하며 각 축은 -110 ~ 110 사이의 값을 가진다.
- 사용자는 X축으로의 이동에 대해 '+' 혹은 '-' 방향으로 제어가 가능해야 한다.
- 사용자는 Y축으로의 이동에 대해 '+' 혹은 '-' 방향으로 제어가 가능해야 한다.
- 사용자는 로봇의 상태를 알 수 있어야 한다. 로봇의 각 축의 위치가 -80 이상이거나 80 이하일 경우는 초록색 LED가, -80보다 작고 -100 이상이거나 80보다 크고 100 이하일 경우는 노란색 LED가, -100보다 작거나 100보다 클 경우는 빨강색 LED가 켜져야 한다.
- 사용자는 이동 속도를 1 ~ 10/sec까지 1/sec 단위로 설정할 수 있어야 한다.
- 사용자는 모든 상태를 초기화할 수 있어야 하며, 초기화 시 X, Y축의 상태는 0이어야 한다.

2-1. 통합기능 시험평가

학습 목표

- 통합기능의 정상적 수행 여부를 판단할 수 있는 시험 절차서를 작성할 수 있다.
- 통합 시험 절차에 따라 단위 소프트웨어와 소프트웨어 컴포넌트를 통합한 프로그램의 운용로직 구현 여부를 평가할 수 있다.
- 통합기능 시험평가 결과를 보고서로 작성할 수 있다.

필요 지식 /

1 통합 시험

1. 통합 시험의 목적 및 정의

통합 시험이란 각 단위 모듈들을 시험한 다음 단계의 시험으로 모듈들이 기능적으로 모여 시스템이나 서브시스템을 형성하게 되는데, 이 과정에서 발생할 수 있는 문제점을 검증하는 것이다.

단위 시험 시에는 시험을 위해 실제 사용되는 코드가 아닌 테스트 드라이버와 스텁을 사용하였으나 이는 가상의 임시적인 것이므로, 모듈을 통합하였을 경우 모듈 간의 상호작용이 제대로 이루어지는지 검사하는 과정이다.

2. 통합 시험 절차

통합 시험을 위해서는 단위 시험 때와 마찬가지로 테스트 진행을 위해 필요한 추가 테스트 케이스를 작성해야 한다. 요구 사항과 설계 문서를 검토하여 코드 분석을 하고 이를 통해 사용되는 변수의 입/출력 정보나 필요한 스텁 정보를 반영하여 작성한다.

통합 시험에서는 시험의 대상 범위와 방법을 설정하게 되는데, 대상 범위는 작게는 모듈 단위부터 크게는 태스크 단위로 묶어서 진행할 수 있다.

이렇게 요구 사항 및 코드 분석 그리고 시험의 대상과 방법을 설정한 이후 이 정보를 바탕으로 테스트 케이스를 작성하고 절차화하고 절차대로 통합 시험을 수행한다.

2 통합 시험 절차서

1. 개요

테스트 목적과 일정, 조직 구성 등을 정의한다.

2. 테스트 케이스

테스트 케이스를 정의한다. 테스트 케이스는 각 시험 대상 소프트웨어의 규모, 성격 그리고 복잡도 등에 따라 작성하는 범위나 내용이 달라질 수 있다. 간단하게 요구 사항 ID, 테스트 케이스 ID, 입출력 데이터 값 정도로 정리할 수도 있고, 각 테스트 케이스마다 테스트용 스크립트까지 작성할 수도 있다.

3. 시험 절차서 샘플 예제

(1) 개요

<표 2-1> 시험 절차서 샘플 - 개요

예제)

1. 개요

1.1 소프트웨어 개요

XXX 소프트웨어는 OO시스템의 각 단계별 점검 및 총조립 점검을 하고 단계별 인터페이스를 모사해 주는 시스템이다.

[그림 X.] 소프트웨어 구성도

1.2 목적

본 SW 시험 절차서는 XXX 소프트웨어의 연동 시험에 대한 개요, 참고 문서, 시험 준비 사항, 시험 내역, 요구 사항 추적성, 참고 사항 등을 규정한 문서이다.

1.3 시험 식별자

기능	기능 구분	비고
소프트웨어 기능	SFR	software functional requirement
인터페이스	IFR	interface requirement
안전 및 보안	SSR	safety & security requirement
환경 및 컴퓨터 자원	ECR	environment & computer resource requirement

(2) 참고 문서: 시험 절차서를 작성하는 데 참고로 사용된 문서

<표 2-2> 시험 절차서 샘플 - 참고 문서

예제)

2. 참고 문서

- 소프트웨어 요구 사항 명세서(SRS, software requirement specification)
- 소프트웨어 설계 기술서(SDD, software design description)
- MIL-STD-498 software development and documentation

문서 번호	문서명
XXX-SRS-XX	XXX 소프트웨어 요구 사항 명세서(SRS, software requirement specification)
XXX-SDD-XX	XXX 소프트웨어 설계 기술서 (SDD, software design description)
XXX-XXX-XX	XXX 소프트웨어 규격서
MIL-STD-498	software development and documentation

(3) 시험 항목

<표 2-3> 시험 절차서 샘플 - 시험 항목

예제)

3. 시험 항목

요구 사항 식별자	요구 사항	시험 식별자	시험명
R-SFR-00X	AAA HW/SW 상태 감시	TC-SFR-00X TC-SFR-00X	AAA 시스템 기동 AAA 시스템 재기동
R-SFR-00X	BBB HW/SW 상태 감시	TC-SFR-00X TC-SFR-00X	BBB 시스템 기동 BBB 시스템 재기동
R-SFR-00X	로그 관리	TC-SFR-00X	로그 관리
R-SFR-00X	경고 및 알림	TC-SFR-00X	텍스트 경고 출력
		TC-SFR-00X	음성 경고 출력
:	:	:	:
R-SFR-0XX	Network 상태 감시	TC-SFR-0XX	네트워크 상태 감시

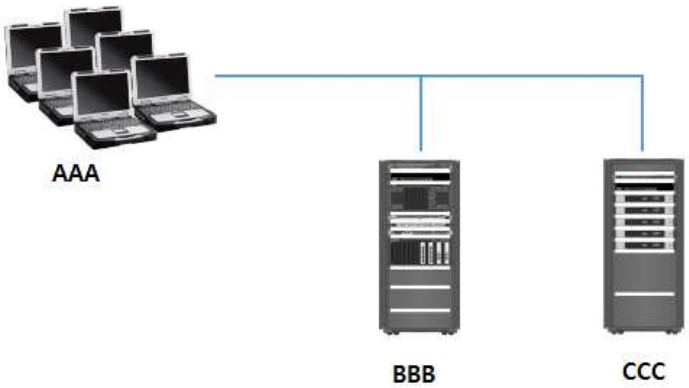
(4) 시험 준비 사항

<표 2-4> 시험 절차서 샘플 - 시험 준비 사항

예제)

4. 시험 준비 사항

4.1 시험 형상



[그림 X.X] XXX 시스템 통합 시험 형상

4.2 하드웨어 준비 사항

구성품	사양	수량	용도
AAA 컴퓨터	Intel i7 이상, RAM 8GB 이상	2ea	AAA SW 탑재용 본장비
BBB	Intel i7 이상, RAM 8GB 이상	1ea	계통별 모니터링
CCC	MVME7100	1ea	제어기 기능 확인
기타	전원 케이블, USB 메모리		

(5) 시험 절차 및 테스트 케이스

<표 2-5> 시험 절차서 샘플 - 시험 절차

예제)

5. 시험 절차

5.1 AAA 시스템 기동

시험명	AAA 시스템 기동	식별자	TC-SFR-00X
설명	AAA 시스템 기동 시 정상적으로 BBB와 연동되는지 점검한다.		
사전 조건	AAA 시험용 컴퓨터에 방화벽 사용 안함으로 되어 있고, SW형상 적용이 되어있는 상태이다.		
절차	1. [BBB SW] 메인 화면에서 상태 감시제어 > 시스템 감시제어 버튼을 클릭한다. (√) 2. [BBB SW] 시스템 감시 윈도우에서 AAA1 의 상태를 확인한다. (√) 3. [AAA] 시스템을 기동하여 XX SW를 실행한다. (√) 4. [BBB SW] 시스템 감시 윈도우에서 XX-AAA1 네트워크 연결 상태를 확인한다. (√) 5. [BBB SW] 시스템 감시 윈도우에서 XX-AAA1 의 상태를 확인한다. (√)		
예상 결과	1. 시스템 감시 윈도우가 전시된다. 2. XX-AAA1 콘솔 아이콘의 테두리(①)가 적색으로 전시된다. 3. XX-AAA1의 SW 들이 자동 시작 된다. 4. 네트워크에 연결된 XX-AAA1의 연결선(①)이 녹색으로 전시된다. 5. XX-AAA1의 모든 상태(CPU 사용량, 디스크 사용량, 메모리 사용량, 프로세스 상태)가 정상이면 콘솔 아이콘의 테두리(①)가 녹색으로 전시된다.		

5.2 AAA 시스템 재기동

시험명	AAA 시스템 재기동	식별자	TC-SFR-00X
설명	AAA 시스템 재기동 시 정상적으로 BBB와 연동되는지 점검한다.		
사전조건	1. BBB 시험용 컴퓨터에 방화벽 사용 안함으로 되어 있고, BBB SW 가 구동중인 상태이다. 2. AAA 시험용 컴퓨터에 방화벽 사용 안함으로 되어 있고, SSI SW가 구동중인 상태이다.		
절차	1. [BBB SW] 메인화면에서 상태 감시제어 > 시스템 감시제어 버튼을 클릭한다. (√) 2. [BBB SW] 시스템 감시 윈도우에서 XX-AAA1 의 상태를 확인한다. (√) 3. [BBB SW] 시스템 감시 윈도우에서 XX-AAA1 네트워크 연결 상태를 확인한다. (√) 4. [BBB SW] XX-AAA1 콘솔 아이콘을 클릭한다. (√) 5. [BBB SW] XX-AAA1 상태 윈도우에서 프로세스 목록 관리(ⓓ)를 확인한다. (√)		
예상 결과	1. 시스템 감시 윈도우가 전시된다. 2. XX-AAA1의 모든 상태(CPU 사용량, 디스크 사용량, 메모리 사용량, 프로세스 상태)가 정상이면 콘솔 아이콘의 테두리(①)가 녹색으로 전시된다. 3. 네트워크에 연결된 XX-AAA1의 연결선(①)이 녹색으로 전시된다. 4. XX-AAA1 상태 윈도우가 전시된다. 5. XX-AAA1 시험용 컴퓨터의 SW 목록과 실행 상태가 전시된다.		

5.3 BBB 시스템 기동

:

:

:

(6) 요구 사항 추적성 매트릭스

<표 2-6> 시험 절차서 샘플 - 요구 사항 추적성 매트릭스

예제)

6. 요구 사항 추적성 매트릭스

소프트웨어 요구 사항		검증 방법				비고
ID	내용	검사	분석	데모	시험	
R-SFR-00X	AAA 시스템을 기동할 수 있어야 한다.			√		
	AAA 시스템을 재기동할 수 있어야 한다.			√		
R-SFR-00X	BBB 시스템을 기동할 수 있어야 한다.			√		
	BBB 시스템을 재기동할 수 있어야 한다.			√		
:	:					

3 VCRM(verification cross reference matrix)

1. 정의

VCRM은 요구 사항과 그 요구 사항을 검증하는 방법을 매트릭스로 표현한 것으로 보통 시스템 통합 시험 시 시스템의 요구 사항을 어떠한 방법으로 검증할 것인가에 대해 시험 계획 시 정의한다. 이때 만일, 요구 사항 명세서가 정의되어 있다면 소프트웨어 요구 사항을 직접 기술하는 대신 SRS(software requirement specification)의 장 절을 인용해도 된다.

2. 검증 방법

(1) 검사(inspection)

기능 항목, 관련 문서, 프로그램 코드 등 정량적으로 판단하기 어려운 항목에 대해 직접 육안으로 확인하여 요구 사항의 만족 여부를 확인하는 방법이다.

(2) 분석(analysis)

분석이란 모델링이나 시뮬레이션 방법, 계산 등 다른 검사 방법을 사용해서 획득한 자료를 분석하여 요구 사항의 만족 여부를 확인하는 방법이다.

(3) 데모(demonstration)

특정한 시험 도구를 사용하지 않고 기능을 직접 수행하여 요구 사항이 만족되는지를 확인하는 방법이다.

(4) 시험(test)

적절한 시험 조건에서 시험 자료를 이용하여 실제로 기능을 수행시켜서 요구 사항이 만족되는지 확인하는 방법이다.

3. 샘플 예제

<표 2-7> VCRM 샘플

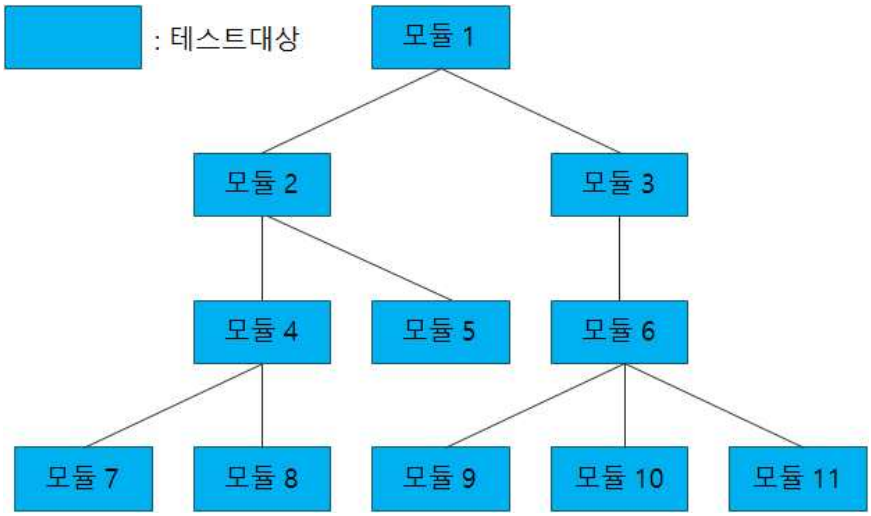
소프트웨어 요구 사항		검증 방법				비고
ID	내용	검사 Inspection	분석 Analysis	데모 Demonstration	시험 Test	
R-01	사용자는 X, Y축의 현재 상태값을 알 수 있어야 하며 각 축은 -110~110사이의 값을 가진다.	V		V		
R-02	사용자는 X축으로의 이동에 대해 '+' 혹은 '-' 방향으로 제어가 가능해야 한다.			V		
:						
:						
R-XX	..			V	V	
:	:					:

4 시험 수행 기법

통합 시험은 모듈을 통합시키는 방법에 따라 여러 가지 테스팅 기법이 있다.

1. 빅뱅 기법

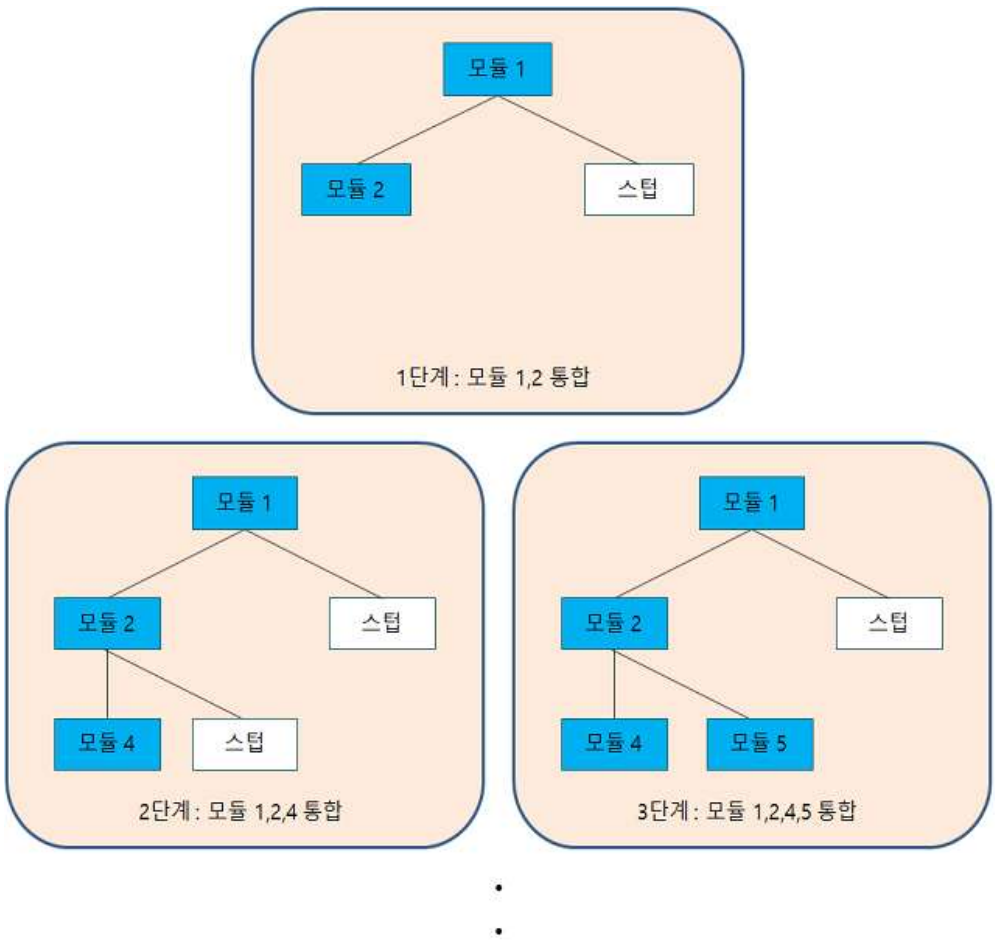
모듈을 한꺼번에 통합하여 테스팅하는 기법으로 가장 빨리 시험할 수 있지만 오류가 날 경우 어느 모듈에서 문제가 생겼는지 확인하기 어려운 단점이 있다. 주로 프로그램의 코드가 별로 크지 않는 경우에 사용한다. 시험하는 절차는 [그림 2-1]과 같다.



[그림 2-1] 빅뱅 기법

2. 하향식(top-down) 기법

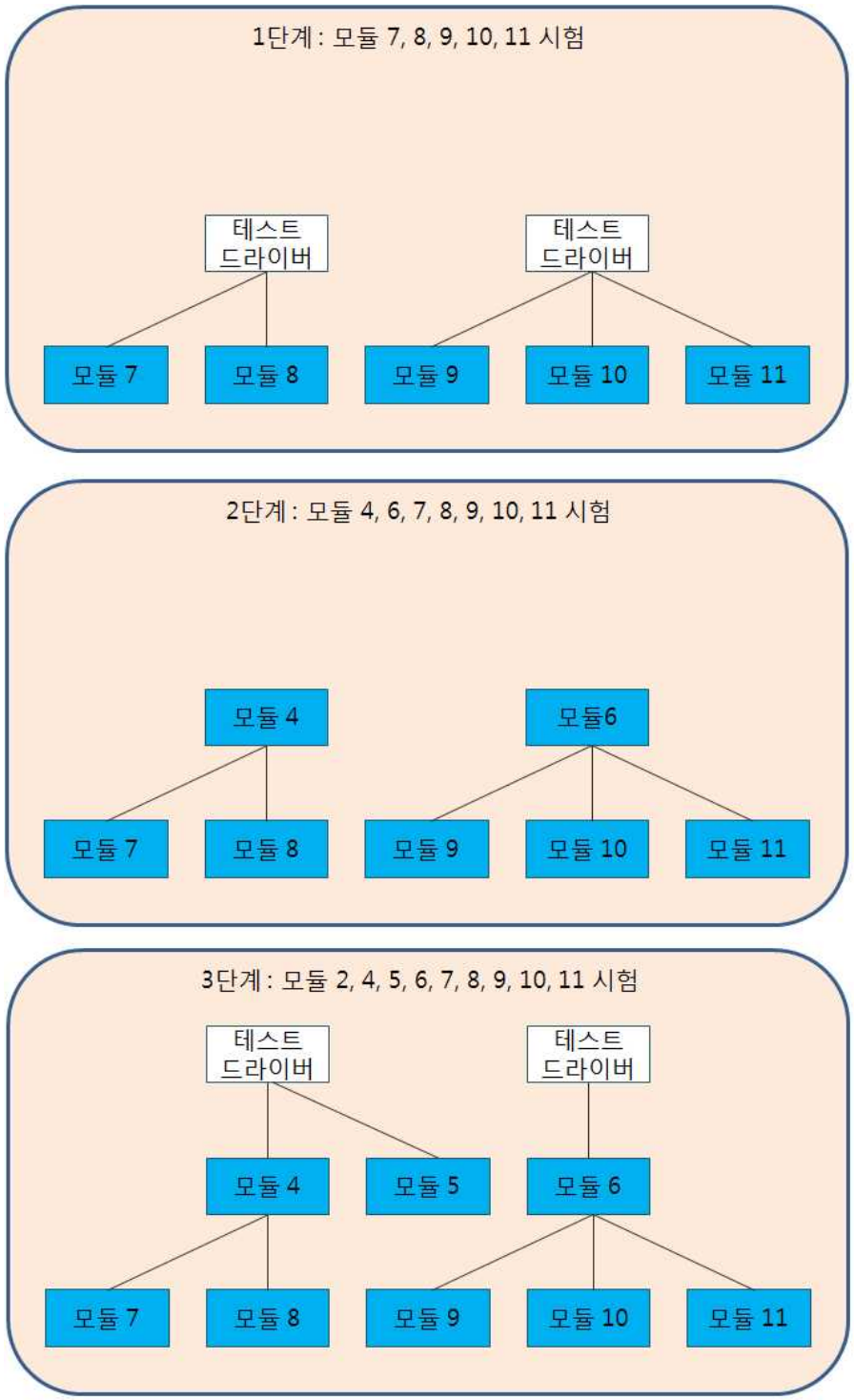
[그림 2-2]와 같이 최상위 모듈부터 하위로 호출되는 모듈을 점진적으로 통합하는 기법으로 상위 모듈 시험 시 하위 모듈에 대한 스텁이 필요하다.



[그림 2-2] 하향식 기법

3. 상향식(bottom-up) 기법

[그림 2-3]와 같이 최하위 모듈부터 상위 모듈로 통합해 가면서 시험하는 방법으로 하위 모듈 시험 시 상위 모듈에 대한 테스트 드라이버가 필요하다.



[그림 2-3] 상향식 기법

5 시험 방법

1. 블랙박스 시험

블랙박스 시험은 내부 구조에 대한 로직보다는 구현되는 기능이나 성능만을 시험한다.

(1) 신택스 시험(syntax testing)

신택스 시험은 입력 데이터가 미리 정의된 조건에 부합하는지 확인하는 기법이다. 예를 들어 제어 속도를 입력받아 처리하는 프로세스의 테스트 케이스를 생각해 보자.

- 입력 변수: 제어 속도
- valid 조건: 0~10
- invalid 조건: 음의 숫자, 10보다 큰 숫자, 문자, 특수문자, 소수점 등

이때 신택스 시험 기법을 적용하면 테스트 케이스는 아래 표와 같이 정의할 수 있다.

<표 2-8> 신택스 시험 기법

순번	입력 유형	입력 값	예상 결과
1	0~10 사이 숫자	5	정상 처리
2	10보다 큰 숫자	180	에러 메시지
3	특수문자	@$	에러 메시지
4	소수점	3.9	에러 메시지

(2) 등가 분할(equivalence patitioning)

등가 분할은 입력값의 범위를 유사한 부분집합으로 나누고, 그 값에 대해 시험하는 방법이다. 시험 조건을 만드는 것이 간단하고 무작위로 샘플링하는 방법보다는 체계적이라 할 수 있다.

로봇의 위치에 따라 알림을 표시해 주는 프로그램의 테스트 케이스는 아래와 같다.

- $-80 \leq$ 위치 ≤ 80: 초록색(G)
- $-100 \leq$ 위치 < -80 or $80 <$ 위치 ≤ 100: 노란색(Y)
- $-100 >$ 위치 or $100 <$ 위치: 빨강색(R)

<표 2-9> 등가 분할 시험 기법

이동속도	-80 ≤ 위치 ≤ 80	-100 ≤ 위치 < -80 80 < 위치 ≤ 100	-100 > 위치 100 < 위치
입력값	25	-89	109
예상 결과값	초록색(G)	노란색(Y)	빨강색(R)
실제 결과값			

(3) 경계값 분석(boundary value analysis)

경계값 분석은 등가 분할과 비슷하나 부분집합의 경계값이나 주위값을 테스트 케이스로 추출하는 방법이다. 위 예를 적용하여 경계값 분석 기법으로 테스트 케이스를 만들면 아래와 같다.

<표 2-10> 경계값 분석 기법

시험 케이스	1	2	3	4	5	6	7	8	9	10	11	12
입력값	-101	-100	-99	-81	-80	-79	79	80	81	99	100	101
예상 결과값	R	Y	Y	Y	G	G	G	G	Y	Y	Y	R
실제 결과값												

(4) 기타

그 외에도 상태전이 테스팅, 유즈케이스 테스팅 등 다양한 기법들이 있다.

6 통합 시험 결과보고서

시험 결과보고서도 시험 절차서나 테스트 케이스를 만드는 것과 마찬가지로 각 프로젝트나 소프트웨어마다 다를 것이다. 중요한 것은 보고서의 목적에 부합되도록 작성하기만 하면 형식에는 크게 구애를 받지 않아도 된다.

시험 결과보고서의 목적은 시험의 결과를 정리하고 이 결과를 바탕으로 소프트웨어의 품질에 대한 평가와 권고 사항 등을 제공하는 것이다.

시험 결과보고서는 반드시 각 시험 단계마다 나올 필요는 없다. 작은 프로젝트 같은 경우는 단위 시험, 통합 시험, 시스템 시험까지 합쳐서 1개의 보고서로 나올 수도 있고, 기간이 길고 규모가 큰 프로젝트의 경우는 매 시험마다 결과보고서를 요청할 수도 있다. 또 단위 시험 결과보고서는 아래 표와 같이 합격/불합격의 결과만 간단하게 정리할 수도 있다.

<표 2-11> 간략한 단위 시험 결과보고서 사례

시험식별자	시험명	합격	불합격	비고
TC-SFR-001	AAA 시스템 기동	√		
TC-SFR-002	AAA 시스템 재기동	√		
TC-SFR-003	BBB 시스템 기동	√		
TC-SFR-004	BBB 시스템 재기동		√	
TC-SFR-005	AAA 시스템 상태 감시	√		
TC-SFR-006	로그 관리	√		
TC-SFR-007	화면 캡처	√		
:	:	√		
:	:			
합 계		00	0	0

full 버전의 정식 시험 결과보고서에는 아래와 같은 항목들로 구성되는 것이 일반적이다.

<표 2-12> 시험 결과보고서 사례

XX 소프트웨어 시험 결과보고서

1. 개요
 1.1 문서 번호
 1.2 적용 범위
 1.3 참고 문서

2. 시험 결과
 2.1 종합
 2.2 세부시험 결과
 2.3 이슈 사항
 2.4 결론 및 권고사항

3. 참고
 3.1 용어 정리
 3.2 문서 변경 절차 및 재개정 이력

수행 내용 1 / 통합기능 시험 절차서 작성하기

재료 · 자료

- 노트 및 필기도구
- 소프트웨어 요구 사항 명세서
- 기타 소프트웨어 설계 산출물 (소프트웨어 설계 명세서, 클래스 다이어그램 등)
- 시험 대상 프로그램(RoboCon) 소스 코드

기기(장비 · 공구)

- 컴퓨터, 윈도우즈 XP 이상
- 사무용 프로그램 (아래한글 or MS Office)
- 프린터
- 인터넷

안전 · 유의 사항

- 시험 절차서에는 요구 사항을 검증할 수 있는 모든 테스트 케이스가 정의되어야 한다.
- 절차서 양식은 크게 구애받을 필요가 없으며 가장 중요한 테스트 케이스만 잘 정리되어 있으면 문제가 없다.

수행 순서

1 요구 사항 및 설계 문서를 파악한다.

학습모듈 1-1에서 작성한 RoboCon 프로그램을 이용하여 요구 조건 및 요구 사항 분석 결과 문서와 클래스 다이어그램 등의 설계 문서를 자세히 검토한다.

2 시험 절차서를 작성한다.

1. 개요를 작성한다.

   소프트웨어 개요, 시험의 목적 등을 기술한다.

2. 참고 문서를 작성한다.

   요구 사항 명세 및 설계 참고 자료를 기술한다.

3. 시험 항목 및 테스트 케이스를 작성한다.

요구 사항 명세 및 설계 참고 자료를 검토한 후 시험 항목과 테스트 케이스를 작성한다.

(1) 시험 항목 식별

시험 항목은 소프트웨어 요구 사항에 나와 있지 않은 항목도 포함될 수 있다. 그러나 요구 사항에 나와 있는 항목은 모두 시험 항목에 포함시켜야 한다.

<표 2-13> 시험 항목 리스트 예제

순번	시험 항목	test case ID	시험 방법
1	프로그램 실행 시 UI가 설계 화면과 같이 도시되어야 한다.	TC-2-01	검사, 데모
2	이동 도시부에 로봇 아이콘이 도시되어야 하며 현재 상태를 반영하여야 한다.	TC-2-02	데모
3	이동 도시부에 현재 로봇의 X, Y축 좌표가 도시되어야 한다.	TC-2-03	데모
4	로봇의 X축의 이동 속도를 0~10 사이 값으로 설정할 수 있어야 한다.	TC-2-04	데모
5	로봇의 X축 이동 방향을 + 혹은 -로 설정할 수 있어야 한다.	TC-2-05	데모
6	로봇의 Y축의 이동 속도를 0~10 사이 값으로 설정할 수 있어야 한다.	TC-2-06	데모
7	로봇의 X축 이동 방향을 + 혹은 -로 설정할 수 있어야 한다.	TC-2-07	데모
8	로봇의 X축의 위치가 -80이상이거나 80이하일 경우는 초록색 LED가, -80보다 작고 -100이상이거나 80보다 크고 100이하일 경우는 노란색 LED가, -100보다 작거나 100보다 클 경우는 빨강색 LED가 켜져야 한다.	TC-2-08	데모
9	로봇의 Y축의 위치가 -80이상이거나 80이하일 경우는 초록색 LED가, -80보다 작고 -100이상이거나 80보다 크고 100이하일 경우는 노란색 LED가, -100보다 작거나 100보다 클 경우는 빨강색 LED가 켜져야 한다.	TC-2-09	데모
:	:	:	:
00	프로그램 종료 버튼을 누르면 프로그램이 정상적으로 종료되어야 한다.	TC-2-XX	데모

모든 시험 항목을 식별하고 위의 표처럼 작성한다. 단, 위의 시험 항목은 하나의 사례일 뿐이다. 필요한 시험 항목을 분류하고 식별하는 것은 시험자의 취향이나 시험 방법에 따라 다르게 작성될 수 있다.

(2) 테스트 케이스를 작성한다.

위에서 식별한 시험 항목에 대해 각각 테스트 케이스를 작성한다. 테스트 방법은 블랙박스 방법으로 수행하며 테스트 케이스 작성 시 테스트 항목의 성격에 따라 신택스 시험, 등가 분할, 경곗값 분석 기법에 따라 절차를 작성한다.

또 상향식 기법으로 시험을 수행할 것인지 하향식 기법으로 수행할 것인지를 결정하고 거기에 맞게 테스트 케이스를 작성한다.

시험 항목 1에 대한 테스트 케이스는 아래와 같이 작성할 수 있다.

<표 2-14> 테스트 케이스 예제1

프로그램명: RoboCon			단계: 통합 시험		
테스트 케이스					
Test Case ID: TC-2-01			Req ID: N/A		
Test Case 설명: 프로그램 실행 시 UI가 설계 화면과 같이 도시되어야 한다.					
테스트 조건: 프로그램이 설치되어 있는 컴퓨터에 로그인 되어 있어야 한다.					
단계	입력값	예상 출력값	실행 결과	Pass/Fail	조치 사항
1	실행	UI 도시			

시험 항목 8에 대한 테스트 케이스는 아래와 같이 여러 경우로 작성할 수 있다. 시험자는 이들 기법 중 하나를 선택할 수도 있고 테스트 케이스를 섞거나 합칠 수도 있다. 테스트 케이스를 어떻게 식별하고 적용하느냐는 전적으로 시험자의 선택이다.

<표 2-15> 테스트 케이스 (등가 분할) 예제2

프로그램명: RoboCon			단계: 통합 시험		
테스트 케이스					
Test Case ID: TC-2-XX			Req ID: R-04		
Test Case 설명: 로봇의 X 축의 위치가 -80이상이거나 80이하일 경우는 초록색 LED가, -80보다 작고 -100이상이거나 80보다 크고 100이하일 경우는 노란색 LED가, -100보다 작거나 100보다 클 경우는 빨강색 LED가 켜져야 한다.					
테스트 조건: 상위 모듈 000 모듈의 테스트 드라이버를 작성해야 한다. 혹은 하위 모듈 000 모듈에 대한 스텁을 작성해야 한다. (시험자 선택)					
단계	입력값	예상 출력값	실행 결과	Pass/Fail	조치 사항

1	25	초록색			
2	-89	노란색			
3	109	빨강색			

<표 2-16> 테스트 케이스 (경곗값 분석) 예제2

프로그램 명 : RoboCon		단계 : 통합 시험			
테스트 케이스					
Test Case ID : TC-2-XX		Req ID : R-04			
Test Case 설명 : 로봇의 X 축의 위치가 -80이상이거나 80이하일 경우는 초록색 LED가, -80보다 작고 -100이상이거나 80보다 크고 100이하일 경우는 노란색 LED가, -100보다 작거나 100보다 클 경우는 빨강색 LED가 켜져야 한다.					
테스트 조건 : 상위 모듈 000 모듈의 테스트 드라이버를 작성해야 한다. 혹은 하위 모듈 000 모듈에 대한 스텁을 작성해야 한다. (시험자 선택)					
단계	입력값	예상 출력값	실행 결과	Pass/Fail	조치 사항
1	-101	빨강색			
2	-100	노란색			
3	-99	노란색			
4	-81	노란색			
5	-80	초록색			
6	-79	초록색			
7	79	초록색			
8	80	초록색			
9	81	노란색			
10	99	노란색			
11	100	노란색			
12	101	빨강색			

(3) 요구 사항 추적성 매트릭스를 작성한다.

시험 항목을 식별하고 테스트 케이스를 작성한 후 요구 사항 추적성 매트릭스를 작성하여 작성한 소프트웨어가 모든 요구 사항을 충족하는지 확인한다. 이는 테스팅 과정 중 확인(verification) 과정의 일부이기도 하다.

소프트웨어 요구 사항을 검증하기 위한 모든 테스트 케이스를 매칭한다.

<표 2-17> 요구 사항 추적성 매트릭스 예

소프트웨어 요구 사항		test case	
ID	내용	ID	시험명
R-01	사용자는 X, Y축의 현재 상태값을 알 수 있어야 하며 각 축은 -110~110 사이의 값을 가진다.	TC-2-01	화면 도시
		TC-2-02	로봇의 현재 상태 도시부에 도시
		TC-2-03	로봇의 X, Y축 좌표 도시
		:	:
R-02	사용자는 X축으로의 이동에 대해 '+' 혹은 '-' 방향으로 제어가 가능해야 한다.	TC-2-00	:
		:	:
R-03	사용자는 Y축으로의 이동에 대해 '+' 혹은 '-' 방향으로 제어가 가능해야 한다.	TC-2-00	:
		:	:
R-04	사용자는 로봇의 상태를 알 수 있어야 한다. 로봇의 각 축의 위치가 -80 이상이거나 80이하일 경우는 초록색 LED가, -80보다 작고 -100이상이거나 80보다 크고 100이하일 경우는 노란색 LED가, -100보다 작거나 100보다 클 경우는 빨강색 LED가 켜져야 한다.	TC-2-08	X축 상태에 따른 LED 표시
		TC-2-09	Y축 상태에 따른 LED 표시
		:	:
R-05	사용자는 이동 속도를 1~10/sec까지 1/sec 단위로 설정할 수 있어야 한다.	TC-2-00	:
		:	:
R-06	사용자는 모든 상태를 초기화할 수 있어야 하며, 초기화 시 X, Y축의 상태는 0이어야 한다.	TC-2-00	:
		:	:
R-07	프로그램 실행 시 CPU 점유율은 최대 20% 사용 메모리는 5MB이하여야 한다.	TC-2-00	:
		:	:

수행 tip

- 프로그램의 규모가 크지 않는 경우 통합 시험은 빅뱅기법을 사용하는 것이 좋다.
- 그러나 학습을 위해서는 상향식 기법 혹은 하향식 기법을 사용해서 통합해가는 과정을 배우는 것이 좋다.

수행 내용 2 / 통합 시험 수행 및 결과보고서 작성하기

재료 · 자료

- 노트 및 필기도구
- 소프트웨어 요구 사항 명세서
- 기타 소프트웨어 설계 산출물 (소프트웨어 설계 명세서, 클래스 다이어그램 등)
- 시험 대상 프로그램(RoboCon) 소스 코드
- 정적 분석 시험 절차서

기기(장비 · 공구)

- 컴퓨터, 윈도우즈 XP 이상
- 사무용 프로그램 (아래한글 or MS Office)
- 프린터
- 인터넷

안전 · 유의 사항

- 공개 소프트웨어를 다운로드하여 사용할 경우 라이선스를 반드시 확인하여 사용 여부를 결정하여야 한다. 일부 소프트웨어는 공개했다가 셰어웨어나 유료 판매 정책으로 수정할 수도 있다.
- 학습을 위해 다운로드한 프로그램은 학습이 완료된 후 학습장의 지침을 확인하여 필요 시 프로그램을 제거해야 한다.

수행 순서

1 시험 절차서를 보고 시험을 준비한다.

앞의 시험 절차서 작성하기의 결과물인 시험 절차서를 확인하고 통합 시험에 필요한 문서 및 장비 등을 준비한다.

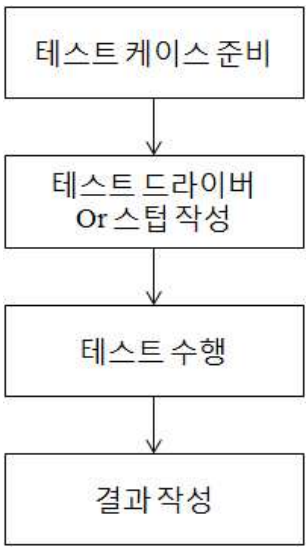
- 노트 및 필기도구
- RoboCon 프로그램

2 통합 시험을 수행한다.

통합 시험의 준비가 완료되었으면, 테스트 케이스에 맞게 시험을 실시한다. 시험의 절차는 테스트 케이스에 준하여 실시하며 1개의 절차가 완료되면 그 결과를 테스트 케이스에 기록한다.

필요 시 테스트 드라이버나 스텁을 작성해서 시험을 해야 한다. 테스트 수행 후 합격/불합격에 대한 결과를 테스트 케이스에 기록하고 시험 결과가 불합격일 경우 자세한 내용 및 프로그래머에게 권고할 사항을 같이 기록한다.

수행 절차는 아래와 같다.



[그림 2-4] 통합 시험 절차

3 시험 결과보고서를 작성한다.

1. '개요'를 작성한다.

시험 대상 프로그램의 설명, 시험 목적, 참고 문서 등을 작성한다.

2. 시험 결과를 작성한다.

(1) 시험 결과 종합표를 작성한다.

시험을 한 결과를 종합하여 아래와 같은 표로 작성한다.

<표 2-18> 시험 결과보고서 종합 사례

시험식별자	시험명	합격	불합격	비고
TC-2-01	화면 도시	V		
TC-2-02	로봇의 현재 상태 도시부에 도시	V		
TC-2-03	로봇의 X, Y축 좌표 도시	V		
:	:			
TC-2-08	X축 상태에 따른 LED 표시	V		
TC-2-09	X축 상태에 따른 LED 표시	V		
:	:			
	합 계	00	0	0

(2) 세부 시험 결과를 작성한다.

각 테스트 케이스별 세부 시험 결과를 작성한다.

(3) 이슈 사항을 작성한다.

시험 결과가 불합격으로 나왔거나 합격이라도 프로그래머에게 알려야 할 사항이 있다면 이슈 사항으로 정리하여야 한다. 이슈 사항은 되도록 자세하게 작성하여야 한다.

(4) 결론 및 권고 사항을 작성한다.

전체 시험 결과에 대한 결론과 권고 사항을 작성한다.

수행 tip

- 신택스 시험, 등가 분할, 경곗값 분석 등의 시험 방법 선택은 시험 항목별로 달라질 수 있으니 적절한 방법을 찾는다.

학습 3 신뢰성 시험평가하기

3-1. 신뢰성 시험평가

학습 목표

- 로봇 소프트웨어 신뢰성 시험평가를 위한 시험 절차서를 작성할 수 있다.
- 시험 절차서에 따라 신뢰성 시험평가 항목에 대한 세부 계획 및 적합한 자동화 시험 도구를 채택할 수 있다.
- 신뢰성 시험평가 항목에 대한 세부 계획 및 적합한 자동화 시험 도구를 채택할 수 있다.
- 자동화 시험 도구를 이용하여 시험평가 항목에 대한 신뢰성 시험을 실시할 수 있다.
- 결과보고서는 산출물 문서에 포함하여 추적성을 유지하도록 작성할 수 있다.

필요 지식 /

① 소프트웨어 신뢰성 시험

소프트웨어 신뢰성이란 주어진 환경에서 주어진 시간 동안 오류 없이 동작할 확률을 말한다. 하드웨어와 달리 소프트웨어에 대한 신뢰성 시험은 정적 분석과 동적 분석을 통해 이루어진다.

1. 정적 분석

정적 분석은 소프트웨어를 실행하지 않은 상태에서 결함을 검출하는 방식으로 사람이 직접 오류를 찾아내는 수동 방식과 자동화 도구를 이용하여 오류 패턴을 찾아내는 자동 방식이 있다.

코드리뷰가 사람에 의해 이루어지는 수동 방식이라면 자동화 도구를 이용하면 코딩 규칙 검증과 실행 시간 오류(runtime error) 검출, 코드의 복잡도 계산, 실행되지 않는 데드코드 등을 검출할 수 있다.

실행 시간 오류(runtime error) 검출 시험은 프로그램의 수행 경로를 추적하면서 프로그램 실행에 영향을 미치는 실행 정지, 오동작, 메모리 누수 등의 런타임 오류를 찾아내는 시험이다. 대표적인 실행 시간 오류의 목록인 CWE-658/659의 목록은 아래 표와 같다.

<표 3-1> CWE-658/659 실행 시간 오류 목록

순번	Defect type	순번	Defect type
1	Buffer Overrun	23	Overlapping Memory Regions
2	Buffer Underrun	24	Redundant Condition
3	Cast Alters Value	25	Return Pointer To Freed
4	Dangerous Function Cast	26	Return Pointer To Local
5	Division By Zero	27	Type Mismatch
6	Double Close	28	Type Overrun
7	Double Free	29	Type Underrun
8	Empty If Statement	30	Uninitialized Variable
9	File System Race Condition	31	Unreachable Call
10	Format String	32	Unreachable Computation
11	Freee Non-Heap Variable	33	Unreachable Condition
12	Free Null Pointer	34	Unreachable Control Flow
13	Integer Overflow of Allocation Size	35	Unreachable Data Flow
14	Ignored Return Value	36	Unreasonable Size Argument
15	Leak	37	Unused Value
16	Misaligned Object	38	Use After Colse
17	Missing Return Statement	39	Use After Free
18	Missing Return Value	40	Useless Assignment
19	Negative File Descriptor	41	Varargs Function Cast
20	No Space For Null Terminator	42	Deadlock
21	Null Pointer Dereference	43	Double Lock
22	Null Test After Dereference	44	Double Unlock

코딩 규칙에 대한 설명은 '2 코딩 규칙' 에서 따로 설명한다.

2. 동적 분석

동적 분석은 소프트웨어를 직접 실행시키면서 검증하는 방법으로 동적 분석의 시험평가 기준은 코드 커버리지(code coverage)이다.

코드 커버리지란 프로그램 코드가 시험을 수행했는가를 백분율로 나타내는 값으로 프로그램의 성격과 종류에 따라서 목표값을 다르게 적용하는 것이 일반적이다. 그리고 동적 분석은 그 특성상 시험을 하는 과정에 비용과 시간이 많이 들어가므로 소프트웨어의 특성과 시험 비용 및 기간을 고려하여 코드 커버리지의 목표값을 적절히 설정해야 한다.

2 코딩 규칙

코딩 규칙이란 코드의 가독성이나 신뢰성을 향상시키기 위해 정한 규칙으로 규칙을 정한 산업 표준 혹은 기관에 따라 코딩 규칙이 조금씩 다르다. 컴파일러가 찾아낼 수 있는 문법적인 오류와 다르다는 면에서는 코딩 스타일과 비슷하나 그와는 달리 정해진 규칙대로 코딩을 수행했는지 걸러내는 툴이 존재하는 것이 코딩 스타일과 다른 점으로 신뢰성을 위해 정해놓은 강제 가이드라인으로 이해하면 된다.

행정안전부에서는 코딩 규칙에 대해 다음과 같이 정의하고 있다. “코드 작성에 관한 지침, 규칙 및 규정의 집합을 말한다. 변수 명명, 코드 들여쓰기, 괄호나 키워드의 위치 정하기 등 주의를 해서 작성하여 하는 문장과 함수, 각 언어의 특성 및 개발환경으로 인해 발생하는 지침을 포함한다. 코딩 규칙은 일반적으로 코드의 가독성, 확정성, 신뢰성을 향상시키기 위한 목적으로 사용한다.”

예를 들어 아래와 같이 C 언어로 작성된 코드를 살펴보자.

```
int v; v++;
printf( “Variable = %d\n” , v);
```

이 코드는 실행 측면에서 보면 문법에 문제가 없어 컴파일도 되고 프로그램 실행에도 문제가 없다. 그러나 몇몇 주요한 코딩 규칙의 관점으로 보면 ‘1줄에 2개의 statement를 썼다는 점’과 ‘변수를 선언하고 초기화 하지 않은 점’의 2가지 규칙을 위배했다고 지적할 것이다.

코딩 규칙은 산업별, 회사별, 기관별로 조금씩 다르게 정해져 있다. 그 중 중요한 코딩 규칙 몇 가지를 살펴보자.

1. MISRA-C, C++

MISRA-C는 자동차산업소프트웨어신뢰성협회(MISRA, motor industry software reliability association)에서 개발된 C 프로그래밍에 대한 개발표준으로 초판은 1998년도에 출간된 MISRA-C:1998 이며, MISRA-C:2004를 거쳐 현재는 MISRA-C:2012까지 출간되었다.

MISRA-C:2012는 총 143개의 룰이 있으며, 각 룰은 '필수', '요구', '권고'로 구분된다. MISRA는 반드시 지켜야 하거나, 지키면 좋은 코딩 규칙으로 이루어져 있는데, MISRA C:2004의 경우는 전체 141개의 코딩 규칙 중에서 반드시 지켜야 할 규칙이 121개, 권고되는 규칙은 20개로 이루어져 있다.

이 표준 문서는 http://www.misra.or.uk 에서 구매할 수 있다.

2. SEI CERT coding standard

전미 소프트웨어협회(software engineering institute)의 CERT(computer emergency response team에서 만든 CERT C 코딩 표준은 수많은 취약성을 분석하여 공통의 프로그래밍 오류를 찾아내고 안전하지 않은 코딩에 대응되는 안전한 코딩 방법을 구축하였다. 이 표준은 89개 규칙과 132개의 권고로 구성되어 있다.

CERT C 코딩 표준에서는 각 분류별로 00-29번은 권고를 30번 이후는 규칙을 나타낸다.

3. CWE-658/659

CWE(common weakness enumeration)는 소프트웨어 보안 및 품질 강화를 위해 개발 시에 참고할 수 있도록 전 세계 SW 취약점을 표준화한 목록으로 미국 국토안전부와 미국 국립표준기술연구소(NIST) 산하 비영리 기관인 MITRE에서 후원하고 있다.

CWE는 CVE 리스트의 다양한 실제 취약성을 기반으로 CWE 커뮤니티의 많은 소스와 예제를 통해 취약성 리스트를 만들어 나간다. CWE-658/659는 MITRE에서 C, C++언어로 소프트웨어 개발 시 공통적으로 취약한 결함을 식별한 목록을 나타낸다.

4. OWASP

OWASP(the open web application security project) 는 오픈소스 웹 애플리케이션 보안 프로젝트이다. 주로 웹에 관한 정보 노출, 악성 파일 및 스크립트, 보안 취약점 등을 연구하며, 10대 웹 애플리케이션의 취약점 (OWASP TOP 10)을 발표했다.

5. DAPA SCR-G 2011 C, C++

한국의 방위사업청에서 만든 코딩 표준으로 신뢰성이 무엇보다 중요한 무기체계의 소프트웨어 개발 시 적용하는 표준으로 개발하였다. 주요한 규칙 몇 가지를 살펴보면 아래와 같다.

(1) 주석 및 명명 규칙

<표 3-2> DAPA의 주석 및 명명 규칙

구분	내 용
주석 규칙	1) C 프로그램 경우 "/* ~ */", C++ 프로그램 경우 "/* ~ */" 또는 "// ~"를 사용한다(단, C 프로그램의 경우 컴파일러가 "// ~"를 지원할 경우에 사용할 수 있다). 2) 주석 내에는 작성자, 날짜, 수정 이력, 기능 설명, 설계 문서와 연계되는 정보 등의 필수 항목은 반드시 포함한다. 3) 주석 내에는 Nested Comment 삽입해서는 안 된다. 4) 기타 주석 작성은 개발자 특성에 맞는 규칙을 사용하되 조직 차원에서

	의 통일된 지침을 수립하여 적용한다. 5) 소스 코드 내 해당 내용을 자세하고 정확하게 설명할 수 있는 주석을 작성하고, 작성 된 주석은 전체 소스 코드의 30% 이상으로 한다.
파일명	가. 함수/클래스일 경우, 파일명을 일치시키고 복수일 경우 파일명은 기능을 잘 설명해 줄 수 있는 명칭으로 한다. 나. 파일의 첫 글자는 대문자 또는 소문자로 통일하여 표기한다. 다. 파일을 잘 설명할 수 있는 이름으로 파일명을 작성한다.
함수/ 클래스/ 메소드명	가. 기능을 잘 표현할 수 있는 명칭을 사용한다. 나. 함수 생성 시 "동사 + (명사)" 순으로 한다. 다. 클래스 생성 시 "ClassName" 형태로 조합하되 첫 단어의 글자는 대문자로 작성한다. 라. 클래스, 함수, 메소드명을 작성할 경우, 실제 기능을 유추할 수 있는 단어를 조합하여 작성한다.
변수명	가. Scope, Linkage를 구분할 수 있어야 하며, 각각 Internal, Extern을 구분하여 작성한다. 예) "Pointer 변수는 p, Global 변수는 g, Static 변수는 s, Count 변수는 n" 으로 표기 나. 변수명의 첫 글자는 반드시 문자를 사용하고 31자를 넘기지 않는다. 다. 변수명은 헝가리언, 카멜, 파스칼 표기법 또는 조직 내부 작성 지침에 따라 사용한다.
매크로명	가. 매크로명은 전체를 대문자 또는 "_" 를 사용한다. 나. 파일, 매크로명은 표준에 있는 예약어를 사용하지 않는다. 예) _변수명, scanf, printf 등

(2) 프로그램 작성 규칙

<표 3-3> DAPA의 프로그램 작성 규칙

구분		주요 내용	개수
C, C++ 공통 적용 (45개)	스타일	한 라인씩 구성, goto문, 다중 return문 자제 등	11
	초기화	변수, 문자열 배열, 포인터 초기화 등	3
	식별자	전역변수 중복 식별자 자제 등	4
	조건식	float 자료형 비교, 조건문 결과 항상 True, False 등	5
	변 환	함수 변환, 데이터형 변환, 상수형 자료 변환 등	8
	포인터/배열	Null 포인터 참조, 배열 인덱스 초과값 사용 등	5
	연산자	0 나누기 연산, side effect, 음수 shift 자제 등	9
C 전용 적용		scanf, malloc, 함수 파라미터 등	5
C++ 전용 적용		throw, virtual function, exception 처리, 생성자, 소멸자 등	15

3 자동화 도구

소프트웨어 신뢰성 시험을 사람이 직접 수행하려면 많은 인력과 비용이 들어갈 뿐만 아니라 실수를 배제할 수 없기 때문에 자동화 도구를 이용하는 것이 도움이 많이 된다. 하지만 테스트 도구를 사용하는 것은 장점뿐 아니라 단점 역시 포함하고 있으므로 테스트 도구에 대한 이해를 통한 적절한 도구를 선택하는 것이 필요하다.

1. 자동화 도구의 장단점

(1) 장점

- 테스트 데이터의 재입력 같은 반복 작업의 자동화
- 테스트 드라이버 및 스텁 같은 테스트 코드 자동 생성
- 코딩 규칙이나 복잡도 계산 등 정의된 평가에 대한 신속하고 정확한 수행
- 성능에 대한 통계와 그래프 등 테스트 정보의 자동 생성 기능

(2) 단점

- 상용 소프트웨어의 경우 투자 비용 및 유지 관리 비용이 높음
- 복잡한 도구의 사용에 대한 학습 필요
- 도입 후 프로세스 적용에 대한 시간, 비용 필요

2. 공개 소프트웨어

요즘은 고가의 상용 소프트웨어뿐만 아니라 테스트 자동화를 위한 공개용 테스트 자동화 도구들이 많이 소개되어 있으니 간단히 살펴보자.

(1) 정적 분석 도구

소스 코드에 대한 코딩 표준/스타일, 복잡도 및 잔존 결함을 발견하기 위하여 사용하는 도구

<표 3-4> 공개용 정적 분석 도구

분류	제품명	세부 정보	
결함 예방/ 발견	pmd(cpd)	개요	자바 소스 코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Linux, Windows
		개발도구 지원	Eclipse, NetBeans 등
		홈페이지	http://pmd.sourceforge.net/
	findbugs	개요	자바 소스 코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Cross-Platform
		개발도구 지원	Eclipse, NetBeans 등
		홈페이지	http://findbugs.sourceforge.net/
	cppcheck	개요	C/C++ 소스 코드에 대한 잠재적인 문제에 대한 분석
		지원 환경	Windows
		개발도구 지원	Eclipse, gedit 등

		홈페이지	http://cppcheck.sourceforge.net/
	sonar	개요	C/C++, Java 등 다양한 언어 지원 및 PMD, CheckStyle 등 플러그인을 통하여 확장 가능
		지원 환경	Cross-Platform
		개발도구 지원	eclipse
		홈페이지	http://www.sonarsource.org/
코딩 표준	checkstyle	개요	자바 프로그램에 대한 코딩 표준 준수 검사 도구로, 다양한 개발도구에 통합하여 활용 가능
		지원 환경	Cross-Platform
		개발도구 지원	Ant, Eclipse, NetBeans 등
		홈페이지	http://checkstyle.sourceforge.net/
	N'SIQ CppStyle	개요	C/C++ 프로그램 언어에 대한 코딩 표준 준수 검사 도구
		지원 환경	Cross-Platform
		개발도구 지원	Visual Studio, eclipse
		홈페이지	http://dev.naver.com/projects/nsiqcppstyle/
	StyleCop	개요	C# 프로그램 언어에 대한 코딩 표준 준수 검사 도구
		지원 환경	Windows
		개발도구 지원	Visual Studio
		홈페이지	http://stylecop.codeplex.com/
코딩 표준	cpplint	개요	구글에서 사용하고 있는 C++ 스타일 가이드 준수 검사 도구로 CLI(파이썬) 형태로 지원됨
		지원 환경	Cross-Platform
		개발도구 지원	-
		홈페이지	http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml#cpplint
코드 복잡도	ccm	개요	소스 코드 복잡도 분석 도구이며, Linux, Mac환경에서는 CLI 형태로 지원됨
		지원 환경	Cross-Platform
		개발도구 지원	Visual Studio
		홈페이지	http://www.blunck.info/ccm.html
	eclipsemetrics	개요	소스 코드 복잡도 분석 소스 코드 통계 정보 제공 도구
		지원 환경	Cross-Platform
		개발도구 지원	Eclipse
		홈페이지	http://www.stateofflow.com/projects/16/eclipsemetrics
	sourcemonitor	개요	윈도우 기반 소스 코드 복잡도 분석 도구
		지원 환경	Windows
		개발도구 지원	-
		홈페이지	http://www.campwoodsw.com/sourcemonitor.html
	cobertura	개요	자바언어에 대한 소스 코드 복잡도 분석 및 커버리지 측정
		지원 환경	Cross-Platform

	개발도구 지원	Ant, Maven
	홈페이지	http://cobertura.sourceforge.net/
javancss	개요	자바언어에 대한 소스 코드 복잡도 분석 도구, CLI 형태로 지원됨
	지원 환경	Cross-Platform
	개발도구 지원	Ant, Jacob
	홈페이지	http://www.kclee.de/clemens/java/javancss/

출처 : http://www.oss.kr/oss_repository6/69515 2016. 08. 05. 검색

(2) 동적 분석 도구

프로그램을 실행하여, 코드 내에 존재하는 메모리 누수, 쓰레드 결함 등을 분석하기 위하여 사용하는 도구

<표 3-5> 공개용 동적 분석 도구

제품명	세부 정보	
Avalanche	개요	Valgrind 프레임워크 기반으로 구현되었으며, 프로그램에 대한 결함 및 취약점 동적 분석 도구
	지원 환경	Linux, Android
	개발도구 지원	-
	홈페이지	http://code.google.com/p/avalanche/
Valgrind	개요	프로그램 내에 존재하는 메모리 및 쓰레드의 결함을 발견하는 동적 분석 도구
	지원 환경	Cross-Platform
	개발도구 지원	Eclipse, NetBeans 등
	홈페이지	http://valgrind.org

출처: http://www.oss.kr/oss_repository6/69515 2016. 8. 5. 검색

(3) 테스트 자동화 프레임워크

단위 테스트, 통합 테스트 등 테스트 단계별 자동화 도구를 활용한 기능 테스트 도구

<표 3-6> 공개용 동적 분석 도구

제품명	세부 정보	
xUnit	개요	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크
	지원 환경	각각의 도구별 지원 환경 상이
	개발도구 지원	eclipse 등

	홈페이지	http://www.junit.org/ http://sourceforge.net/apps/mediawiki/cppunit/index.php?title=Main_Page http://www.nunit.org/
STAF	개요	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	eclipse
	홈페이지	http://staf.sourceforge.net/
FitNesse	개요	웹기반 테스트 케이스 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	eclipse
	홈페이지	http://fitnesse.org/
NTAF	개요	NHN 테스트 자동화 프레임워크이며, STAF와 FitNesse를 통합
	지원 환경	Cross-Platform
	개발도구 지원	eclipse, Maven 등
	홈페이지	http://dev.naver.com/projects/ntaf/
Selenium	개요	다양한 브라우저 지원 및 개발언어를 지원하는 웹 애플리케이션 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://seleniumhq.org/
watir	개요	Ruby 기반 웹애플리케이션 테스트 프레임워크
	지원 환경	Cross-Platform
	개발도구 지원	-
	홈페이지	http://watir.com/

출처 : http://www.oss.kr/oss_repository6/69515 2016. 08. 05. 검색

4 코드 커버리지

100 % 코드 커버리지란 의미는 모든 소스 코드를 한 번 이상 시험을 수행했음을 의미하는 것이다. 그러나 복잡한 프로그램의 경우 100 % 코드 실행률을 달성하기란 매우 어려우며 또 코드 실행을 했다고 해서 에러가 없는 것은 아니다.

코드 커버리지는 여러 가지 종류가 있다.

1. 문장 커버리지(statement coverage)

문장 커버리지는 말 그대로 코드 문장의 수행 여부를 판단한다. 코드의 수행 여부만을 판

단하기 때문에 조건문에서는 항상 True만 존재하는지 Flase인 경우도 존재하는지 확인되지 않는다. 아래 코드에서는 4번째 Statement(ss=1;)가 실행되지 않는다.

```
int Test(int a, int b) {
    int ss=0;
    if(a==1 && b==1) {
        ss=1;
    }
    return ss;
}
```

테스트 케이스를 작성하기가 매우 쉬우나 보장성이 가장 낮다. 이 경우 테스트 케이스는 한 가지이면 충분하다.

2. 분기 커버리지(branch coverage)

분기 커버리지는 조건문이 있을 경우 그 조건문에 True와 False인 경우가 각각 존재하는지를 검출한다.

그러므로 위 코드의 조건문 'if(a==1 && b==1)' 은 전체 4가지 경우 중 Result가 False인 3가지 경우를 모두 하나의 False로 검출하므로 조건문 내부 각각의 조건에 대한 커버리지는 측정되지 않는다. 그러므로 아래 표의 3, 4번은 실행하지 않는다.

<표 3-7> 분기 커버리지의 테스트 케이스

번호	a==1	b==1	Result
1	True	True	True
2	True	False	False
3	*False*	*True*	*False*
4	*False*	*False*	*False*

테스트 케이스 2가지로 커버리지를 채울 수 있다.

3. 변형다중조건 커버리지(modified condition/decision condition)

MD/DC는 조건문 내에서도 각각의 조건문의 수행 여부를 검출하며, 프로그램의 수행 과정에서 True와 False가 각각 존재하는지 검출한다.

MC/DC 조합 결정 방법은 아래와 같다.

- 모든 결정 포인트 내의 전체 조건식은 적어도 한 번 모든 가능한 결과값(True, False)을 갖도록 한다.

- 결정 포인트 내의 모든 개별 조건식은 적어도 한 번 모든 가능한 결과값(True, False)을 갖도록 한다.
- 결정 포인트에 있는 각각의 개별 조건식은 다른 개별 조건식에 영향을 받지 않고 그 결정 포인트의 결과값에 독립적으로 영향을 준다.

각 조건문 내에서 가능한 결과값을 가지려면 Result와 a, b가 각각 True와 False가 한 번씩 나오면 된다. 따라서 4번 조건은 실행하지 않고 테스트 케이스 3가지가 나온다.

<표 3-8> MC/DC의 테스트 케이스

번호	a==1	b==1	Result
1	True	True	True
2	True	False	False
3	False	True	False
4	*False*	*False*	*False*

수행 내용 1 / 신뢰성 시험 절차서 작성하기

재료 · 자료

- 노트 및 필기도구
- 소프트웨어 요구 사항 명세서
- 기타 소프트웨어 설계 산출물 (소프트웨어 설계 명세서, 클래스 다이어그램 등)
- 시험 대상 프로그램(RoboCon) 소스 코드

기기(장비 · 공구)

- 컴퓨터, 윈도우즈 XP 이상
- 사무용 프로그램 (아래한글 or MS Office)
- 프린터
- 인터넷

안전 · 유의 사항

- 시험 절차서 작성 시 시험의 순서를 잘 나열하여야 한다. 그렇지 않으면 필요 없는 절차를 재수행하는 등 비효율적으로 될 수 있다.

수행 순서

1 시험의 개요와 참고 자료를 작성한다.

'학습 1 단위기능 시험평가' 시 작성하였던 RoboCon 프로그램을 이용하여 정적 분석과 동적 분석을 수행하는 시험의 개요를 작성한다.

개요에는 문서명과 프로그램명, 시험의 목적 등을 기록한다.

참고 자료에는 요구 사항과 설계 문서 그리고 단위 시험 결과보고서, 통합 시험 결과보고서 등을 기록한다.

2 시험 준비 사항을 기록한다.

시험 준비 사항에는 신뢰성 시험을 다음 위한 시험 형상, 준비해야 할 소프트웨어, 하드웨어 목록, 시험 장비 등을 기록한다.

준비해야 할 소프트웨어 목록에는 정적 분석용 툴 CPPCheck를 포함시킨다.

3 신뢰성 시험 절차를 작성한다.

1. 시험의 종류와 방법을 결정한다.

정적 분석으로는 코딩 규칙 검증 및 실행 시간 오류 분석, 동적 분석으로는 코드 커버리지 확인 시험을 추가한다.

2. 코딩 규칙

코딩 규칙 검증은 'DAPA SCR-G 2011 C, C++'를 적용하도록 하고 자동화 도구 없이 검사자가 코드를 직접 보면서 수행하도록 한다.

3. 실행 시간 오류 분석

실행 시간 오류 분석은 공개 소프트웨어인 자동화 도구를 사용하여 수행하도록 계획한다. 자동화 도구는 CPPCheck를 사용한다.

4. 코드 커버리지 목표값 설정

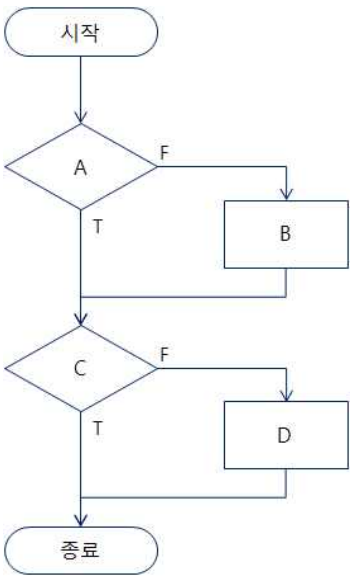
코드 커버리지 확인을 위한 공개 소프트웨어는 대개 통합개발환경(IDE)에 플러그인 형태로 설치되기 때문에 Visual Studio나 Eclipse같은 개발환경이 갖추어지지 않은 컴퓨터에서 수행하기가 어렵다. 검사자가 코드를 직접 확인 및 로깅을 하면서 수행하도록 계획한다.

코드 커버리지의 목표값은 아래와 같이 설정한다.

- 문장 커버리지: 100 %
- 분기 커버리지: 100 %

4 테스트 케이스를 작성한다.

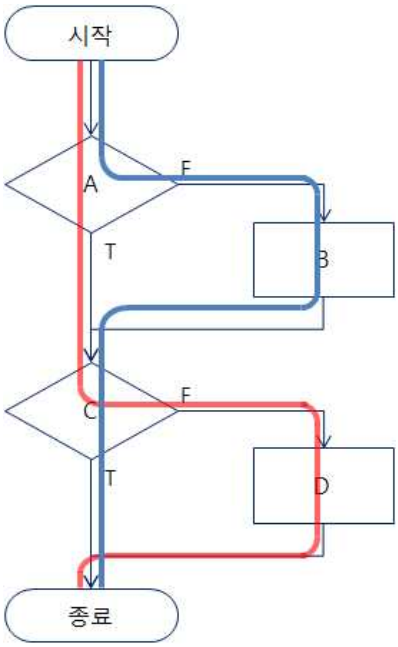
테스트 케이스를 작성하기 전 코드 커버리지의 목표를 세운다. RoboCon 프로그램은 그리 규모가 크지 않으니 분기 커버리지 100 %를 목표로 테스트 케이스를 작성한다.



[그림 3-1] 프로그램 흐름 (예)

만일, 프로그램의 흐름이 위와 같이 되어 있고 분기 커버리지 100 %를 달성하기 위해서는 테스트 케이스를 2개를 작성해야 한다.

- 테스트 케이스 1: 시작 → A(T) → C(F) → D → 종료 (파란색)
- 테스트 케이스 2: 시작 → A(F) → C(T) → D → 종료 (빨강색)



[그림 3-2] 분기 커버리지 테스트 케이스

테스트 케이스를 작성하기 위해서는 시험 대상 프로그램에 대해 클래스 다이어그램과 같은 설계 문서와 프로그램 로직, 코드의 흐름을 정확히 이해해야 한다. 또 조건문과 분기문 등을 확인하여 제어 흐름을 만들고 각 분기점에서 True와 False의 경우를 모두 통과하도록 테스트 케이스를 설계하여야 한다.

마지막으로 모든 테스트 케이스를 검토하여 분기 커버리지 100 %가 될 수 있도록 테스트 케이스가 다 작성되었는지 확인한다.

수행 내용 2 / 정적 분석 수행하기

재료 · 자료

- 노트 및 필기도구
- 소프트웨어 요구 사항 명세서
- 기타 소프트웨어 설계 산출물 (소프트웨어 설계 명세서, 클래스 다이어그램 등)
- 시험 대상 프로그램(RoboCon) 소스 코드
- 정적 분석 시험 절차서

기기(장비 · 공구)

- 컴퓨터, 윈도우즈 XP 이상
- 사무용 프로그램 (아래한글 or MS Office)
- 프린터
- 인터넷

안전 · 유의 사항

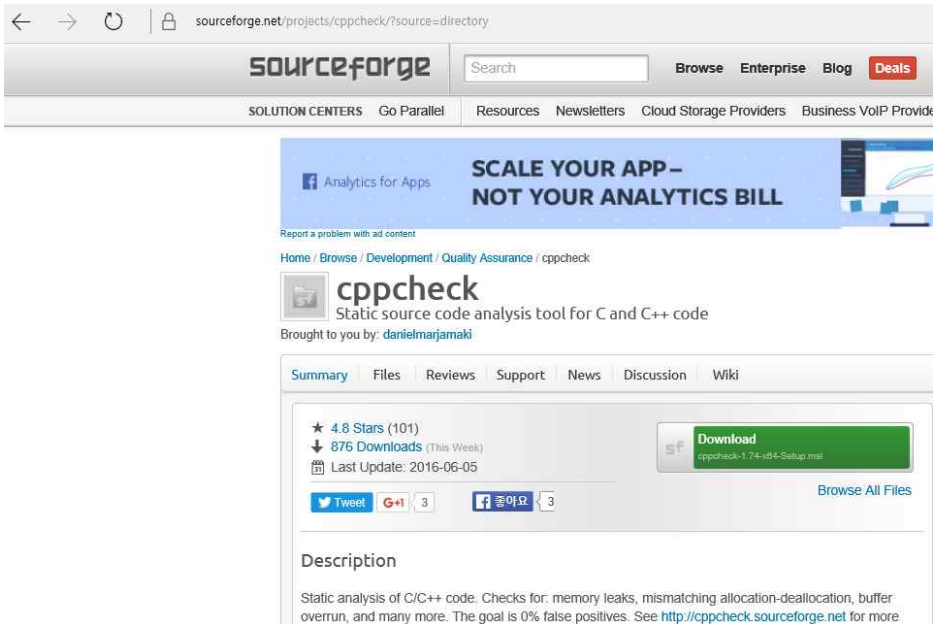
- 공개 소프트웨어를 다운로드하여 사용할 경우 라이선스를 반드시 확인하여 사용 여부를 결정하여야 한다. 일부 소프트웨어는 공개했다가 셰어웨어나 유료 판매 정책으로 수정할 수도 있다.
- 학습을 위해 다운로드한 프로그램은 학습이 완료된 후 학습장의 지침을 확인한 후 필요 시 프로그램을 제거해야 한다.

수행 순서

1 정적 분석용 공개 소프트웨어를 설치한다.

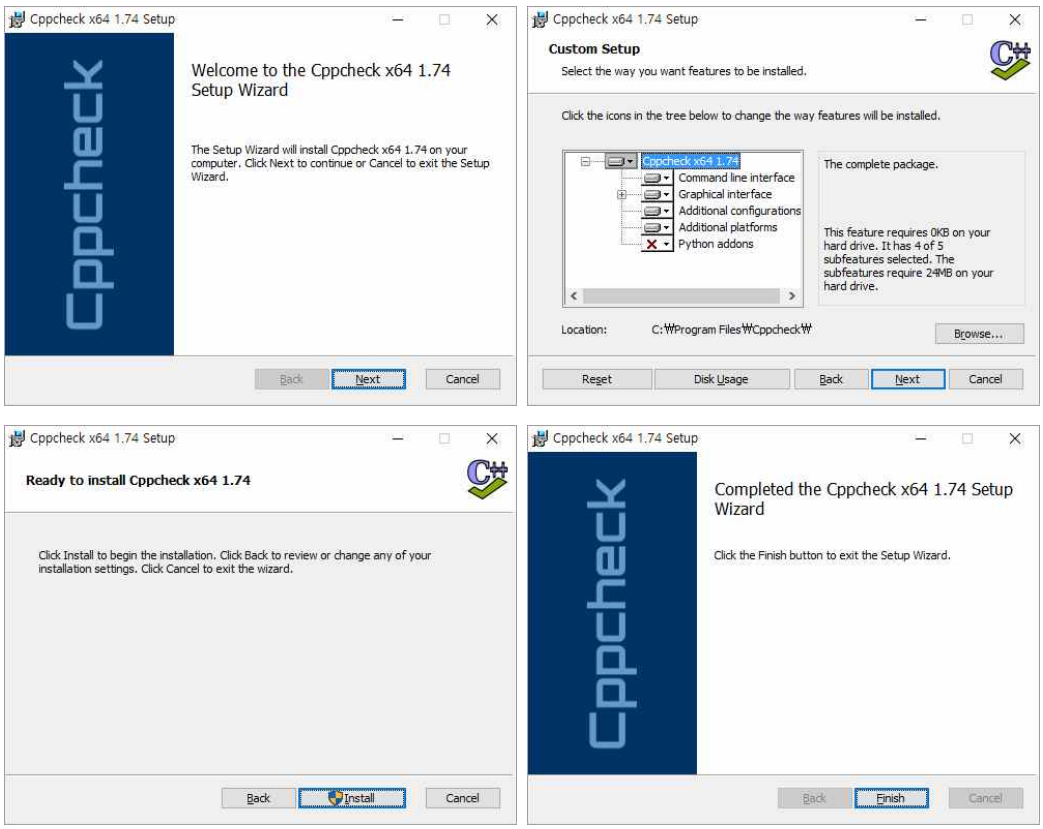
정적 분석을 수행하기 위해서는 정적 분석용 툴이 있어야 하며, 대상 소프트웨어가 있어야 한다. 정적 분석용 툴은 공개 소프트웨어 중 CPPCheck를 다운로드하여 사용하며 대상 소프트웨어는 '학습 1 단위기능 시험평가' 시 작성하였던 RoboCon 프로그램을 이용한다.

1. 아래 웹사이트로 이동하여 CPPCheck를 다운로드한다.
   (http://cppcheck.sourceforge.net/)



출처 : http://cppcheck.sourceforge.net/ 2016. 08. 02. 스크린샷.
[그림 3-3] CPPCheck 다운로드 화면

2. 다운로드한 파일을 설치한다.



[그림 3-4] CPPCheck 설치 과정

3. 프로그램 설치가 안 될 경우 온라인으로 검사한다.

만일, 프로그램을 설치를 할 수 없다면 아래의 사이트로 이동하여 온라인으로 코드를 검사할 수도 있다. 단, 온라인 데모를 이용할 경우 사용에 불편함이 있다.

(http://cppcheck.sourceforge.net/demo/)

2 예제 프로그램 작성하여 정적 분석한다.

CPPCheck 도구의 간단한 사용법과 정적 분석을 수행하는 과정을 확인하기 위해 간단한 샘플 프로그램을 작성하여 실행해본다.

1. 예제 프로그램을 작성한다.

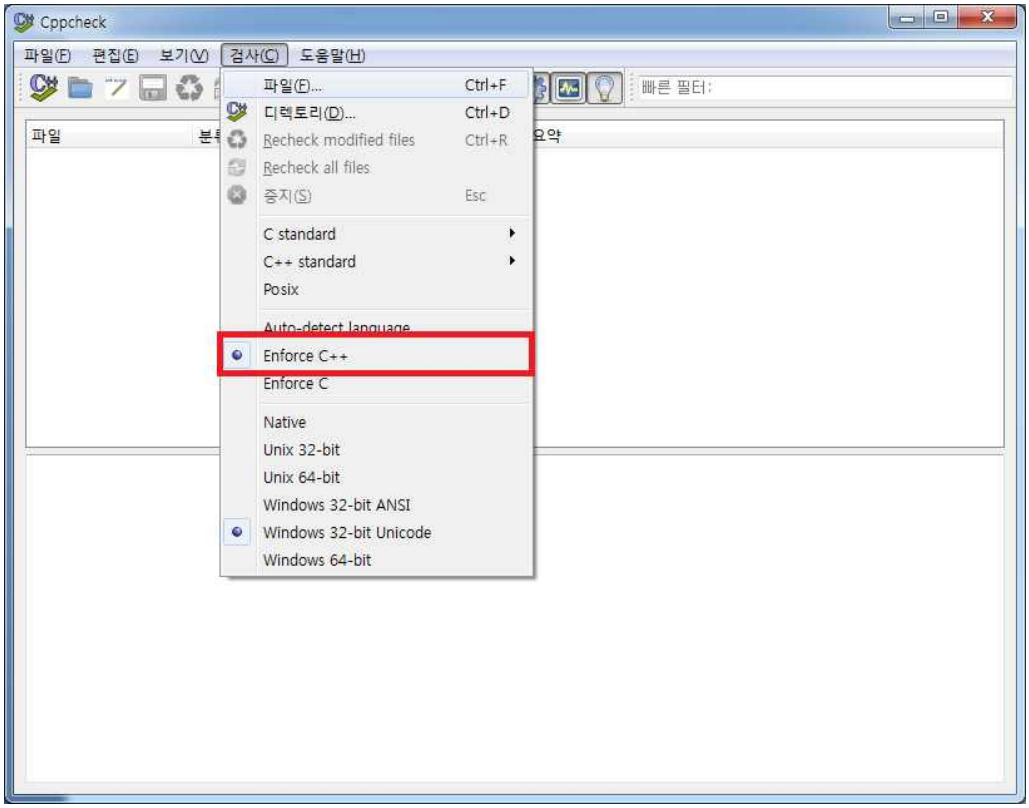
메모장을 열어 아래와 같이 코딩한 후 file1.c로 저장한다.

```
void aaa(char* str)
{
   char* buf =new char[8];
   strcpy(buf, str);

   FILE* file = fopen("sample.txt","w");
   if(!file)
      return;

   for(char* c=buf; *c; ++c)
      fputc((int)*c, file);

   delete buf;
}
```

2. CPPCheck 프로그램으로 분석한다.

(1) CPPCheck 프로그램을 열고 아래와 같이 설정한다.

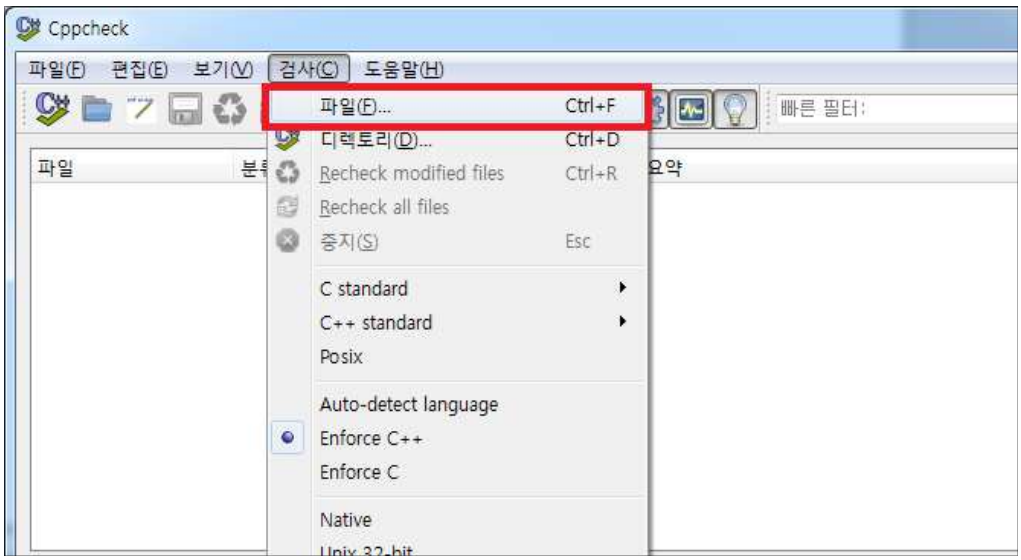
검사 메뉴를 선택 후 'Enforce C++'을 선택한다. 이것은 검사 시 프로그램 언어를 C++ 기준으로 검사한다는 뜻이다. 프로그램 언어를 C로 설정하여 검사를 하면 결과가 달라질 수 있다.



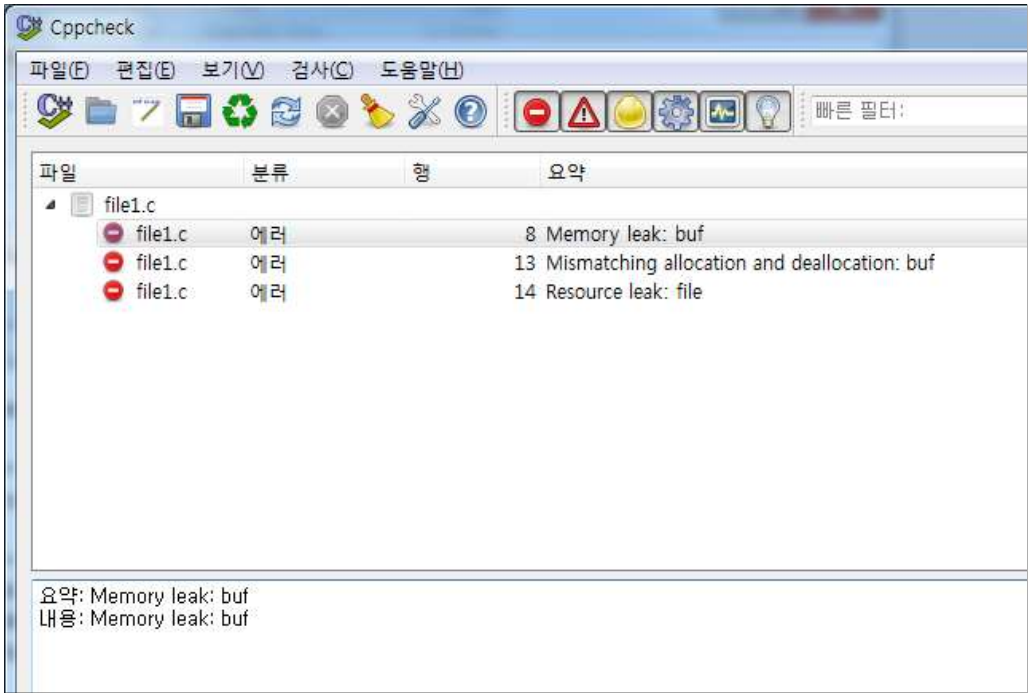
[그림 3-5] CPPCheck 프로그램 언어 설정

(2) 검사를 수행한다.

검사-파일을 선택하여 저장된 file1.c 파일을 연다. 파일을 열면 바로 검사를 수행하고 그 결과를 아래 창에 도시하여 준다.



[그림 3-6] 검사-파일 메뉴



[그림 3-7] 검사 결과 - 오류 발견

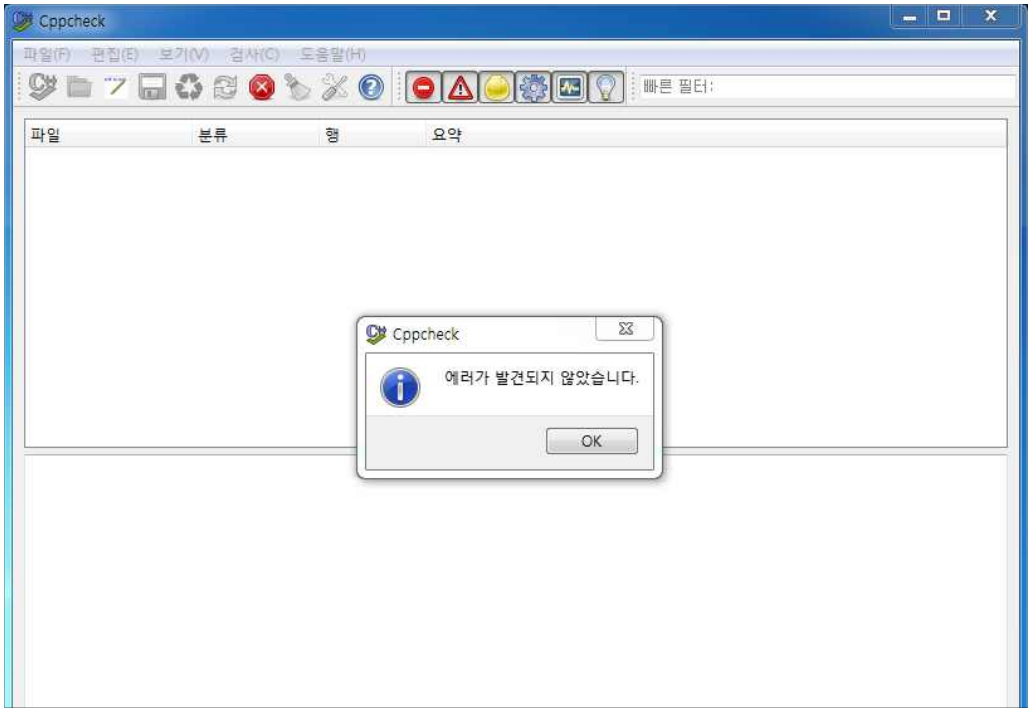
에러가 있는 각 항을 선택하면 아래 창에 에러의 내용을 보여준다.

CPPCheck 가 검출하는 에러는 아래와 같다.

- 메모리 및 리소스 Leaks
- Out of Bound 체크
- 사용하지 않는 클래스의 Private Function 체크
- 표준 템플릿 라이브러리(STL)에서 허용하지 않는 사용 체크
- 초기화하지 않은 변수 체크
- 기타

(3) 오류를 수정한다.

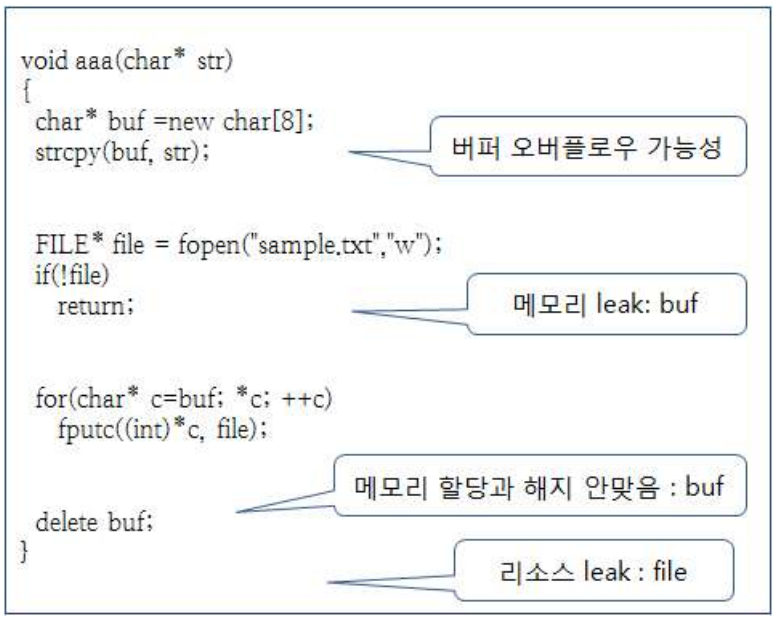
각 오류 사항을 확인하고 수정한다. '검사-Recheck all files' 메뉴를 선택하고 재검사하여 아래와 같이 '에러가 발견되지 않았습니다.' 라는 메시지가 뜨면 모든 오류가 수정된 것이다.



[그림 3-8] 검사 결과 - 에러 없음

(4) 코드리뷰를 통해 자동 검사 툴의 결과와 비교한다.

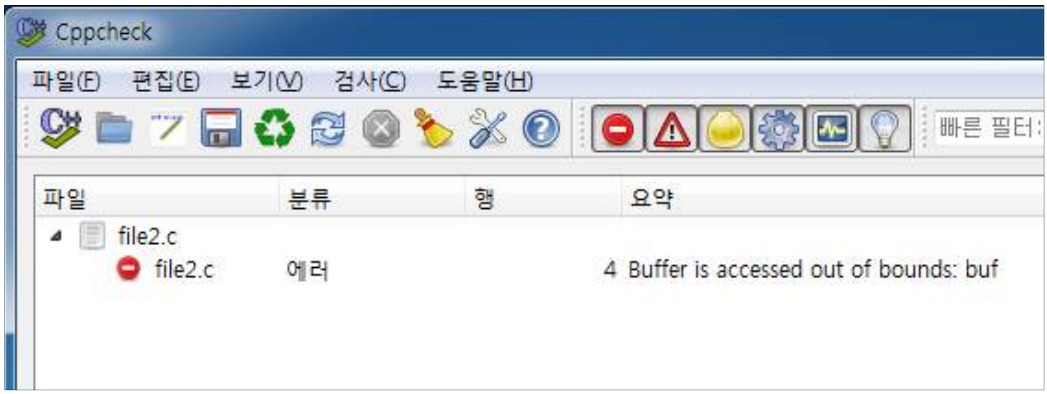
file1.c 코드를 자세히 살펴보면 4곳에서 문제가 있음을 확인할 수 있다.



그러나 CPPCheck는 이 중 버퍼 오버플로우 가능성은 체크를 못하고 3곳의 오류만을 찾아냈다.

다음 코드를 검사해보자.

```
void main(char* str)
{
  char buf[8];
  strcpy(buf, "Hello world!");
  printf("%d\n", *buf);
}
```



[그림 3-9] file2.c 에 대한 오류

CPPCheck가 'Out of Bound' 에러가 났음을 보여준다.

(5) 자동 검사 툴의 한계를 확인한다.

위 두 사례에서 보는 바와 같이 같은 에러 사항 '버퍼 오버플로우' 에 대해서 경우에 따라 에러를 검출해낼 수도 있고 아닐 수도 있다. 따라서 자동 검사 툴로 검사를 해서 오류가 없다고 해서 무결점의 코드라고 할 수 없음을 검사자는 인식하여야 한다.

3 RoboCon 프로그램을 정적 분석한다.

학습모듈 1, 2에서 작성 및 검사를 수행했던 RoboCon 프로그램에 대해 정적 분석한다.

1. CPPCheck 도구를 열고 설정을 한다.

검사 메뉴를 눌러 하위 메뉴에서 운영체제 환경과 작성한 프로그램의 언어를 선택한다.

2. 프로그램을 검사한다.

'검사-파일' 을 선택해서 각각의 소스 파일을 검사해도 되고 '검사-디렉토리' 를 선택해서 한꺼번에 프로그램을 구성하는 모든 파일을 검사해도 된다.

3. 에러를 기록하고 수정한다.

검사 결과 에러가 검출되면 우선 그 사항을 기록한다. 결함 상태는 인지, 해결, 무시로 나눌 수 있다.

- 인지: 결함의 내용과 위치를 확인한 상태
- 해결: 코드를 수정하여 결함을 해결한 상태
- 무시: 도구는 결함이라고 검출했으나 프로그래머의 의도 하에 만들어진 코드로 수정이 필요 없는 상태

<표 3-9> 에러 기록표 사례

순번	파일명	결함 위치	결함 유형 및 내용	결함 상태
1	file1.c	8	Memory leak: buf	인지
2	file1.c	13	Mismatching allocation and deallocation:buf	인지
3	file1.c	14	Resource leak: file	인지
:	:	:	:	

이후 에러를 확인하고 수정한다. 에러가 나오는 줄을 더블클릭하면 소스가 열려 편집 가능하다. 에러 수정 후에는 반드시 컴파일러(DEV-C++)에 넣어 컴파일 및 실행을 한 후 다시 검사한다. 최종적으로 에러가 없어질 때까지 이 과정을 반복한다. 결함 상태 모두 해결 혹은 무시로 되어 있어야 한다.

4. 코드리뷰를 수행한다.

도구로 검사할 수 없는 부분이나 프로그램 로직 상의 오류는 검사자가 코드와 설계 자료를 보면서 검출해 내야 한다. 이 부분까지 마쳐야 정적 분석을 완료했다고 할 수 있다.

4 정적 분석 결과보고서를 작성한다.

정적 분석 완료 후 정적 분석 결과보고서를 아래와 같이 작성한다.

<표 3-10> 정적 분석 결과보고서 예시

XX 소프트웨어 정적 분석 결과보고서

1. 요약

정적 분석 구분	사용 도구/방법	결함 개수	조치율
코딩 규칙 검증	코드리뷰(동료검토)	2	100%
실행 시간 오류	CPPCheck	10	100%

2. 상세

순번	파일명	결함 위치	결함 유형 및 내용	결함 상태
1	file1.c	8	Memory leak: buf	해결
2	file1.c	13	Mismatching allocation and deallocation:buf	해결
3	file1.c	14	Resource leak: file	해결
:	:	:	:	

3. 기타

수행 내용 3 / 동적 분석 수행하기

재료 · 자료

- 노트 및 필기도구
- 소프트웨어 요구 사항 명세서
- 기타 소프트웨어 설계 산출물 (소프트웨어 설계 명세서, 클래스 다이어그램 등)
- 시험 대상 프로그램(RoboCon) 소스 코드
- 정적 분석 시험 절차서

기기(장비 · 공구)

- 컴퓨터, 윈도우즈 XP 이상
- 사무용 프로그램 (아래한글 or MS Office)
- 프린터
- 인터넷

안전 · 유의 사항

- 공개 소프트웨어를 다운로드하여 사용할 경우 라이선스를 반드시 확인하여 사용 여부를 결정하여야 한다. 일부 소프트웨어는 공개했다가 세어웨어나 유료 판매 정책으로 수정할 수도 있다.
- 학습을 위해 다운로드한 프로그램은 학습이 완료된 후 학습장의 지침을 확인하여 필요 시 프로그램을 제거해야 한다.

수행 순서

1 동적 시험을 준비한다.

동적 시험은 코드를 실행하면서 동작의 오류를 검사하는 행위이므로 코드를 실행하기 위한 환경을 갖추어야 한다. 만일, 자동화 도구가 있다면 자동화 도구를 실행시키고 시험을 위한 세팅을 해야 한다.

1. 시험 절차서를 검토한다.

동적 시험을 수행하기 위해 제일 먼저 해야 할 일은 시험 절차서를 검토하고 시험 환경을 갖추는 것이다. 시험 절차서에는 시험에 필요한 하드웨어, 소프트웨어, 시험에 필요한 지원 장비 등이 명시되어 있으니 이와 함께 기록을 위한 도구 등을 준비한다.

2. DEV-C++ 프로그램을 실행한다.

RoboCon 프로그램 소스를 읽어들이기 위해 프로그램 개발 시 사용하였던 DEV-C++를 실행하고 RoboCon 프로그램 소스를 불러온다.

[2] 동적 시험을 실시한다.

1. 시험 수행할 테스트 케이스를 검토한다.

시험 절차서의 순서대로 첫 번째 테스트 케이스를 열어 테스트를 수행할 모듈과 입출력 값, 절차 등을 확인한다.

2. 필요한 경우 테스트 드라이버 혹은 스텁을 작성한다.

각 모듈에 대한 테스트를 하기 위해서는 그 모듈을 불러들이기 위한 테스트 드라이버나 혹은 그 모듈이 호출하는 다른 모듈을 대신해주는 스텁을 작성해야 한다.

3. 테스트 케이스를 수행하고 결과를 기록한다.

1개의 테스트 케이스 내에 있는 절차대로 테스트 케이스를 수행하고 그 결과값과 합부 판단을 아래와 같이 기록한다.

<표 3-11> 테스트 케이스 예제

프로그램 명 : RoboCon			단계 : 단위 시험		
테스트 케이스					
Test Case ID : TC-1-XX			Req ID : N/A		
Test Case 설명 : 함수명 BIT_Out(Res) A 입력값과 B 입력값을 확인하여 BIT 테스트 결과를 내주는 모듈					
테스트 조건 : 상위 모듈 000 모듈의 테스트 드라이버를 작성해야 한다.					
단계	입력값	예상 출력값	실행 결과	Pass/Fail	조치 사항
1	A=1, B=1	Res=1	*1*	*Pass*	*없음*
2	A=0, B=1	Res=0	*0*	*Pass*	*없음*

이러한 절차대로 각 테스트 케이스를 시험 후 결과를 기록한다. 모든 조건문에 대해 True와 False의 경로를 다 거쳤는지 확인하여야 한다.

3 결과보고서를 작성한다.

동적 분석 완료 후 결과보고서를 아래와 같이 작성한다.

<표 3-12> 동적 분석 결과보고서 예시

XX 소프트웨어 동적 분석 결과보고서

1. 요약

순번	분류	결과	비고
1	문장 커버리지	100%	N/A
2	분기 커버리지	95%	Unreachable Code 있음.

2. 상세

순번	클래스명	함수명	테스트 케이스 수	성공수	문장 커버리지
1	AAA	A_aa	4	4	100%
2	AAA	A_bb	2	2	100%
3	BBB	B_aa	3	3	100%
:	:	:	:		

3. 기타

수행 tip

- 학습 내용 1 단위기능 시험평가하기의 테스트 케이스를 참조로 작성한다.