

---

# 차 례

---

학습모듈의 개요	1
학습 1. 로봇 미들웨어 사양 분석하기	
1-1. 로봇 미들웨어 사양 분석	3
• 교수·학습 방법	17
• 평가	18
학습 2. 로봇 미들웨어 설계하기	
2-1. 로봇 미들웨어 설계	21
• 교수·학습 방법	37
• 평가	38
학습 3. 로봇 미들웨어 구현하기	
3-1. 로봇 미들웨어 구현	41
• 교수·학습 방법	55
• 평가	56
참고 자료	59



# 로봇 미들웨어 설계 학습모듈의 개요

## 학습모듈의 목표

로봇 어플리케이션 개발자가 로봇의 다양한 센서, 액추에이터 및 공통 기능 요소들을 사용하기 쉽게 추상화하여 API로 제공하고 이들을 운용 및 관리할 수 있는 로봇 미들웨어를 개발할 수 있다.

## 선수학습

로봇 공학, 메카트로닉스, 운영체제론

## 학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 로봇 미들웨어 사양 분석하기	1-1. 로봇 미들웨어 사양 분석	1903080304_14v1.1	로봇 미들웨어 사양 분석하기
2. 로봇 미들웨어 설계하기	2-1. 로봇 미들웨어 설계	1903080304_14v1.2	로봇 미들웨어 설계하기
3. 로봇 미들웨어 구현하기	3-1. 로봇 미들웨어 구현	1903080304_14v1.3	로봇 미들웨어 구현하기

## 핵심 용어

미들웨어, 로봇 운영 체제, 시스템 아키텍처, 소프트웨어 아키텍처, 로봇 소프트웨어 계층 구조

## 학습 1 로봇 미들웨어 사양 분석하기

학습 2 로봇 미들웨어 설계하기

학습 3 로봇 미들웨어 구현하기

### 1-1. 로봇 미들웨어 사양 분석

#### 학습 목표

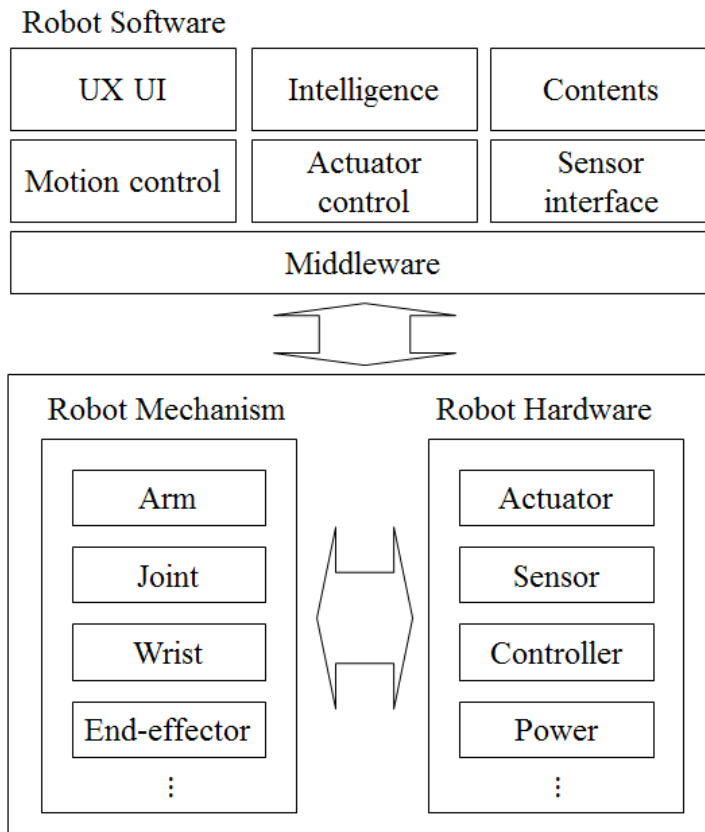
- 로봇 요구 사항에 따르는 소프트웨어 운영 체제가 제공하는 서비스를 분석할 수 있다.
- 로봇 요구 사항에 따르는 하드웨어를 구동하기 위한 통신 프로토콜을 분석할 수 있다.
- 로봇 요구 사항에 따르는 로봇 소프트웨어를 분류하고 공통 요소들을 도출할 수 있다.
- 소프트웨어 공통 요소 및 응용프로그램 간 상호 정보 교환 및 프로세스 관리를 위한 미들웨어 구조 사양을 도출할 수 있다.

#### 필요 지식 /

##### ① 미들웨어(middleware)

###### 1. 미들웨어 정의

일반적으로 미들웨어는 응용 소프트웨어(application software) 운영 체제(operation system, OS)로부터 제공 받는 서비스 이외에, 추가적인 서비스를 제공하는 컴퓨터 소프트웨어라고 정의한다. 여기서 응용 소프트웨어는 유연하고 확장 및 축소가 편리하며, 이러한 장점을 충족하기에 개발자의 다른 기종 간 플랫폼을 다시 구축할 필요가 없어야 한다. 예로 데이터베이스 시스템, 전자 통신 소프트웨어, 메시지 및 쿼리 처리 소프트웨어를 들 수 있다. 미들웨어는 양쪽을 연결하여 데이터를 주고받을 수 있도록 중간에서 매개 역할을 하는 소프트웨어, 혹은 네트워크를 통해서 연결된 여러 개의 컴퓨터에 있는 많은 프로세스들에게 어떤 서비스를 사용할 수 있도록 연결해 주는 소프트웨어로 정의할 수도 있다. 최근에는 인터넷의 일반화와 융복합 기술의 발전으로 인해 어떤 것을 특정 지어 미들웨어라 정의하기 힘든 상황이다. 즉, 미들웨어라는 기술이 광범위하게 확장되어지고 양쪽을 이어주는 중간적인 기능을 담당하는 소프트웨어를 모두 미들웨어라고 얘기할 수 있는 시기이다. 특히, 인터넷을 기반으로 하는 통신 기능의 확대를 통하여 서버 컴퓨터와 클라이언트 컴퓨터를 연결하는 소프트웨어도 미들웨어의 한 종류라고 얘기할 수 있다. 또 로봇 시스템에서 로봇을 운영하는 운영 소프트웨어와 장애물 혹은 제품을 인식하는 비전 센서 응용 프로그램 사이에서 두 소프트웨어에 필요한 정보나 기능을 제공해 주는 소프트웨어 역시 미들웨어라고 말할 수 있다.



[그림 1-12] 로봇 시스템의 구성

로봇 소프트웨어에서 미들웨어를 사용하는 이유는 운영 소프트웨어와 미들웨어 혹은 미들웨어를 포함하는 운영 소프트웨어를 사용하면 로봇을 개발하는 과정에서 시간, 효율성 및 신뢰성을 얻을 수 있기 때문이다. 로봇 소프트웨어 개발 과정에서 엄청나게 많은 소프트웨어를 작성해야 하는 과정을 미들웨어를 사용함으로 상당히 많이 줄일 수 있다.

또 안정적으로 개발되어 있는 소프트웨어를 사용함으로써 소프트웨어 개발 과정의 효율성을 높이며 동시에 소프트웨어의 신뢰성 또한 높일 수 있다. 기존의 개발되어 있는 미들웨어 소프트웨어를 활용함으로써 응용프로그램에서는 추가적인 소프트웨어 개발이 필요 없게 되어 소프트웨어 개발의 효율성을 높이게 된다. 동시에 기존의 개발되어 있는 미들웨어 소프트웨어를 활용함으로써 응용프로그램과 미들웨어의 호환에 문제가 없다면 소프트웨어의 신뢰성도 높일 수 있게 되는 것이다.

대표적인 예로, 안드로이드 운영 체제를 사용하여 핸드폰을 개발하는 핸드폰 개발 회사이다. 핸드폰 개발 회사는 짧은 시간에 안정적이며 신뢰성 있는 제품을 개발하기 위해서 안정적인 안드로이드 운영 체제(미들웨어 포함)를 선택하여 제품을 개발하여 개발 기간을 줄이고 효율성 및 신뢰성을 향상시키는 것이다.

<표 1-2> 미들웨어 사용에 따른 시스템의 영향

번호	기능	미들웨어 사용 시	미들웨어 미사용 시
1	네트워크 및 시스템 부하 관리	<ul style="list-style-type: none"> <li>- 데이터 압축 기술을 이용하여 필요한 서비스 요청 및 결과 데이터만 전송</li> <li>- 작업의 중요도에 따른 부하의 분산이 가능</li> </ul>	<ul style="list-style-type: none"> <li>- 많은 쿼리 문장 및 선택된 모든 데이터가 전송</li> <li>- 별도의 부하 관리 도구가 필요</li> </ul>
2	응용프로그램 변경 관리	<ul style="list-style-type: none"> <li>- 서버 단위로 변경 관리</li> </ul>	<ul style="list-style-type: none"> <li>- 클라이언트 단위로 변경 관리</li> </ul>
3	분산 트랜잭션 관리	<ul style="list-style-type: none"> <li>- 분산된 업무를 동시에 처리 가능(자료의 일관성)</li> </ul>	<ul style="list-style-type: none"> <li>- 분산된 업무를 순차적 처리로 동시 처리가 불가능(자료의 불일치성 내제)</li> </ul>
4	보안 기능	<ul style="list-style-type: none"> <li>- 네트워크 데이터의 암호화 및 서비스의 종류에 따른 사용자 통제 관리</li> </ul>	<ul style="list-style-type: none"> <li>- 데이터가 네트워크에 개방되어 보안에 취약</li> <li>- 서비스별 사용 통제 등의 다양한 보안 미 지원</li> </ul>
5	상호 운용성 및 확장성	<ul style="list-style-type: none"> <li>- 다양한 환경 지원</li> <li>- 체계가 다른 업무와 상호 연동이 가능</li> </ul>	<ul style="list-style-type: none"> <li>- 동일 DB간 연동 가능(대규모 데이터 연동 시 성능 저하)</li> </ul>
6	응용프로그램 복잡도	<ul style="list-style-type: none"> <li>- 클라이언트와 서버의 2개 프로그램 개발</li> </ul>	<ul style="list-style-type: none"> <li>- 클라이언트 프로그램만 개발</li> </ul>

출처: 전준식(2014). 『거침없이 배우는 JBoss』. (주)지앤선. 12

## 2. 미들웨어 종류

### (1) 데이터베이스 접근 미들웨어(database access middleware)

응용프로그램의 SQL(structured query language) 요청과 데이터베이스 서버 사이의 일련의 접근 방법을 이용하여 보다 쉬운 데이터베이스의 접근을 제공한다. 일반적으로 데이터베이스 접근 미들웨어 산업표준에는 JDBC와 ODBC가 있으며 네트워크 접속되어 있는 데이터베이스 서버와의 접속, 데이터베이스 정보 보안 및 일관된 자료의 인터페이스를 갖추고 있다.

### (2) 트랜잭션 프로세싱 모니터(transaction processing monitor)

트랜잭션(최소한의 과업 수행)이 프로세서 내의 한 단계로부터 다음 단계로 잘 넘어가는 것을 관리하는 프로그램이다. 이의 목적은 트랜잭션 처리가 완전하다는 것을 보장하고, 만약 오류가 발생하면 적절한 조치를 취하기 위함이다. 트랜잭션 프로세싱 모니터는 부하 분산을 채용한 3계층 구조에서 특히 중요한데, 한 트랜잭션이 여러 대의 서버 중 어느 곳으로도 전달될 수 있기 때문이다. 실제로, 많은 모니터가 모든 부하 분산 동작을 처리하고 트랜잭션을 그들의 유용성에 따라 다른 서버로 보내고 있다.

(3) 분산 컴퓨터 환경(distributed computing environment)

분산 컴퓨터 환경은 다양한 플랫폼 간의 분산 컴퓨팅을 위한 OSF(open software foundation)에서 개발한 산업계 표준으로 분산 컴퓨터들의 시스템 내에서 컴퓨팅 및 데이터 교환을 설정하고 관리하는 데 필요한 산업표준 소프트웨어 기술이다. 일반적으로 분산 컴퓨터 환경은 여러 가지 다른 크기의 서버들이 지리적으로 퍼져 있는 대형 컴퓨터 시스템의 네트워크 내에서 클라이언트 및 서버 모델을 이용하여 구축된다. 분산 컴퓨터 환경이 지원하는 기능은 DCE RPC, DCE 이름 서비스, DCE 시간 서비스, DCE 보안 서비스, DCE 분산 파일 시스템, DCE 쓰레드 등이 있다.

(4) ORB(object request broker)

ORB는 객체 간의 클라이언트 서버 관계를 맺어주는 미들웨어이다. ORB를 사용하면 클라이언트는 서버 객체에 있는 메서드를 자유롭게 투명하게 호출할 수 있다. ORB는 호출을 가로채어 요구를 처리할 객체를 찾고, 매개변수를 전달하고, 메서드를 호출하고 또 처리 결과를 돌려주는 일 등을 담당한다. 클라이언트는 객체 인터페이스를 제외하고는 객체의 위치나 해당 객체를 개발한 프로그램 언어, 운영 체제, 그 밖에 시스템과 관련된 어느 것도 알 필요가 없다. ORB는 이질적인 분산 환경에서 서로 다른 컴퓨터 내에 이질적인 응용프로그램 간의 상호 운용성과 연결성을 제공한다.

(5) 원격 프로시저 콜(remote procedure call)

원격 프로시저 콜은 한 프로그램이 네트워크상의 다른 컴퓨터에 있는 프로그램에 서비스를 요청하는 데 사용되는 프로토콜 시스템으로 이때 서비스를 요청하는 프로그램은 네트워크에 대한 상세 내용을 알 필요가 없다. 원격 프로시저 콜은 원격에서 서로 다른 플랫폼에서 제공되는 데이터 및 서비스를 이용하는 분산 컴퓨팅 환경의 핵심 구성 요소로 포함되듯이 다른 미들웨어의 하부 시스템으로 포함되는 경우가 대다수이다.

(6) 메시지 지향 미들웨어(message-oriented middleware)

메시지 관리를 위한 미들웨어로 일반적으로 큐(queue)를 이용한 응용프로그램 사이의 메시지 관리 미들웨어이다. 응용프로그램 사이의 비동기 비연결형 기술을 사용하여 통신 개선 수단을 제공하는 통신 기반의 미들웨어이다. 여기서 통신 대화는 마치 두 사람이 일련의 전자 우편 메시지를 교환하는 것과 매우 유사하게, 응용프로그램 사이에서 주고받는 다수의 메시지로 구성될 수 있다. 이 메시지들은 임시 저장소(queue)로 보내어져 수신 측에서 가져갈 때까지 보관되어 축적 전송 방식으로 사용한다. 예로 한 사용자가 전자 상거래 서버에 주문 요구를 보낸다고 가정하면 이 요구는 즉시 처리될 수도 있고 임시 저장소에서 처리가 될 때 까지 대기할 수도 있다. 심지어 일부 요구 사항들은 퇴근 후에 처리가 될 수도 있다.

## ② 로봇 운영 체제(operating system)

### 1. 로봇 운영 체제와 미들웨어

최근에 소프트웨어 운영 체제에 대한 필요성이 증대되면서 많은 분야에 전용 운영 체제가 개발되어 활성화되고 있다. 대표적인 사례가 안드로이드로 대표되는 핸드폰 운영 체제이다. 핸드폰을 비롯한 많은 제품의 소프트웨어를 개발하는 과정에서 제품 개발의 기간을 단축시키고 효율적이면서 신뢰성 있는 소프트웨어를 개발하기 위해서 운영 체제를 많이 활용하고 있다. 또 기존에 개발된 운영 체제는 기초적인 프로그램 실행이나 메모리 관리에서 확대되어 네트워크 운영 및 사용자 인터페이스까지 확대되고 있다. 이와 같이 기존의 운영 체제 기능의 확대는 네트워크 기능을 포함하는 미들웨어 기능으로 확대되고 있다. 특히, 전 세계에서 로봇 소프트웨어를 개발하는 과정에 운영 체제를 경쟁적으로 개발하여 로봇 산업을 선점하려고 하고 있다. 따라서 로봇 미들웨어 개발 과정을 로봇 운영 체제를 중심으로 미들웨어를 포함하여 기술하여, 미들웨어와 함께 로봇 운영 체제에 대한 이해를 높이도록 하겠다.

### 2. 로봇 운영 체제

#### (1) ROS(robot operating system)

ROS는 2007년 5월 미국의 스탠퍼드 대학 인공지능 연구소(AI LAB)가 진행하는 STAIR (stanford ai robot) 프로젝트를 위해 MORGAN QUIGLEY이 개발한 SWITCHYARD라는 시스템에서 시작하였다. 2007년 11월 미국의 로봇 전문 기업 WILLOW GARAGE사가 이어 받아 ROS라는 이름으로 개발하기 시작하였다. WILLOW GARAGE사는 개인 로봇 (personal robot) 및 서비스 로봇 분야에서 매우 유명한 회사이다. 그리고 우리가 알고 있는 영상처리 오픈소스인 OpenCV, 키넥트(kinect)와 같은 3차원 기기에 많이 사용되고 있는 PCL(point cloud library)을 개발하고 지원했던 것으로도 유명했다.

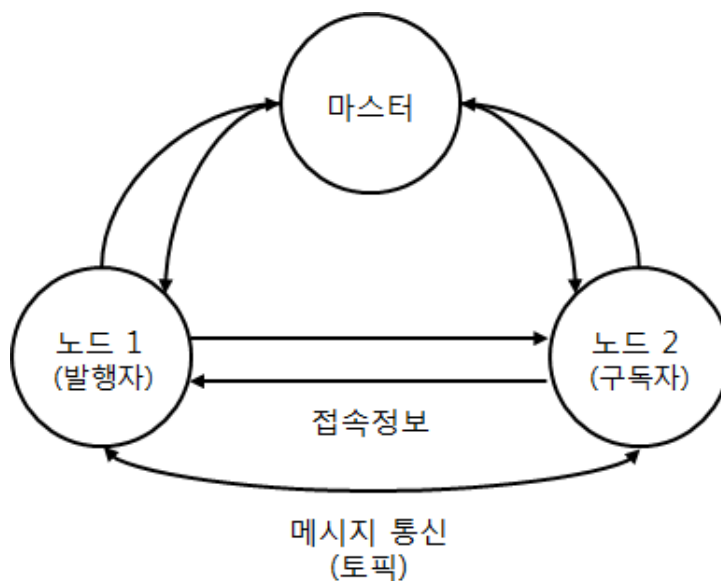
WILLOW GARAGE사는 2007년 11월부터 ROS 개발에 착수하여 2010년 1월 ROS 1.0이 세상에 나왔다. 우리에게 공식적으로 알려진 버전은 ROS BOX TURTLE이라는 이름으로 2010년 3월에 소개되었다. 그 뒤에도 C TURTLE, DIAMONDBACK 등 우분투나 안드로이드와 비슷하게 ABC 순으로 버전을 작명하고 있다. 2014년 7월에는 ROS 9번째 버전 ROS 인디고 이글루(INDIGO IGLOO, 공식적으로 8번째 버전)를 오랜 베타 과정을 거쳐 공개하였다.

ROS는 BSD 라이선서(BSD 3-clause licence)를 기반으로 하고 있어서 누구든지 수정, 재사용, 재배포가 가능하고 로봇 관련 학회를 통해 먼저 알려지기 시작하였다. 특히, ROSDay와 ROSCon이라는 개발자와 사용자를 대상으로 하는 콘퍼런스가 열리고 있으며 ROS Meetup이라는 이름으로 다양한 커뮤니티 모임도 진행되고 있다. 그뿐만 아니라 ROS를 적용할 수 있는 로봇 개발도 빠르게 이루어지고 있다.



ROS는 하드웨어 추상화, 저수준 장치 제어, 공통적으로 사용 가능한 기능, 프로세스 간의 메시지 전달, 패키지 관리와 같은 로봇 소프트웨어 운영에 필요한 다양한 기능들을 제공한다. ROS는 프로세싱 노드와 내부 구성 파일, 각종 라이브러리 등을 패키지라는 형태로 묶어서 로봇 소프트웨어 모듈을 구성하는 기본 단위로 배포한다. ROS는 그래프와 노드 관리, 분산 서비스 개발과 같은 여러 가지 개발 지원 도구들을 제공하고 있지만, 통합된 형태의 개발 환경이 아닌 개별 도구들의 집합 형태이기 때문에 사용자들이 쉽게 접근하기 어려운 측면이 있다. 현재 주로 리눅스 운영 체제를 지원하고 있고 윈도우 지원은 미약한 실정이다.

ROS는 그래프 기반 아키텍처를 채택하고 있으며 그래프는 노드들의 연결로 구성된다. 노드는 연산을 수행하는 프로세스로서, 하나의 로봇 시스템은 일반적으로 많은 수의 노드로 구성된다. ROS에서 각 노드는 [그림 1-2]와 같이 발행자/구독자(publish / subscribe) 방식의 토픽(topic)이라는 중간 매개체를 통해 센서 정보나 제어 명령, 상태 정보 등의 메시지를 다른 노드에게 보내거나 받을 수 있으며, 요청/응답(request / reply) 방식의 원격 서비스 호출 기능을 제공한다.



[그림 1-2] ROS 메시지 통신 절차

ROS는 각 노드를 위한 명시적인 컴포넌트 모델을 갖고 있지 않고 토픽과 서비스 호출 기능을 통해 프로세스 간 통신 기능을 제공하는 IPC(inter-process communication)을 제공하는 형태이다. 따라서 노드들이 다른 노드와 어떤 형태로 연결되어 있는지 그 연결 관계가 각 노드 프로그램에 하드 코딩되어 있어서 컴포넌트를 조합하는 시스템 설계 시점에 전체 구성을 알기 어렵고, 바이너리 형태의 코드가 수행되어야 그 연결 관계를 확인할 수 있다.

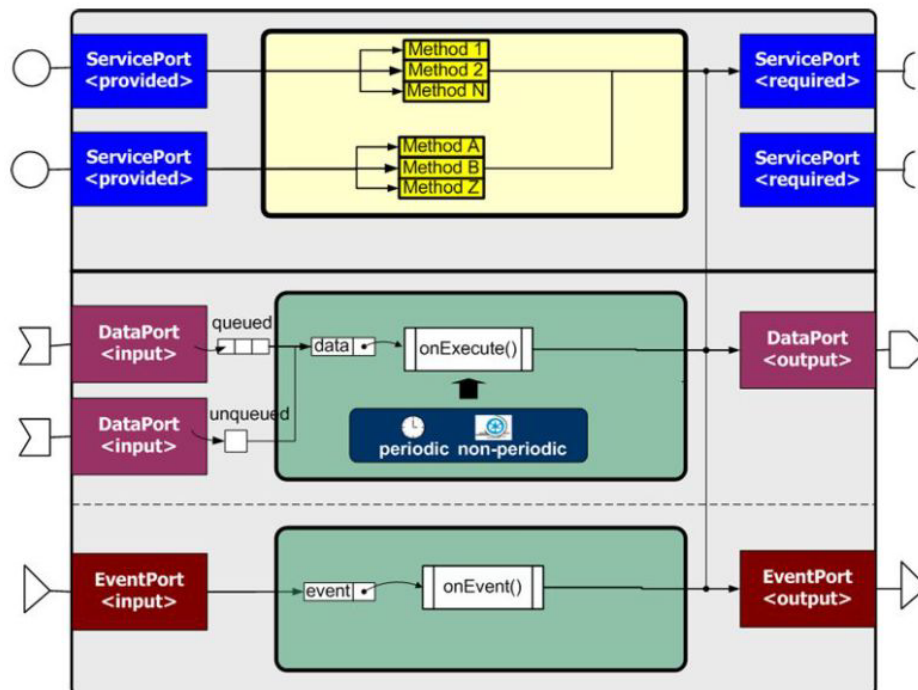
## (2) OPRoS(open platform for robotics services)

2000년대 국가에서 로봇 소프트웨어의 중요함을 인식하여 로봇 소프트웨어 개발에 관심을 가지고 정식으로 로봇용 플랫폼을 개발하기 시작하였다. 이후 산업자원부 SPIRE(S/W platform initiative for robotics engineering)와 정보통신부 RUPI(robot unified platform initiative)로 나뉘어 개별적으로 추진되었다. 2008년 새 정부 출범에 따라 로봇 산업 정책이 지식경제부로 옮겨지게 되었고, 이때 OPRoS라는 이름하에 하나의 로봇 소프트웨어 플랫폼 개발이 시작되었다. 2009년 10월부터 소스코드를 전부 공개하여 현재까지 계속 업데이트가 진행 중이다.

OPRoS가 추구하고자 하는 목적은 로봇용 플랫폼과 통합된 개발 환경을 제공하여 다양한 콘텐츠를 제공하고 로봇 산업 발전에 더욱 박차를 가하는 데 있다. 실제로 애플사는 iOS기반의 iPod, iPhone을 출시하면서 MAC OS에서 Objective-C기반의 MAC용 어플리케이션을 개발하는 Xcode 개발도구를 통해 개발 환경과 시뮬레이터를 제공하였고 iPhone 쇼크라는 세계적인 사회현상을 만들기도 하였다. 이에 한발 더 나아가 구글은 Android를 오픈소스로 제공하고 통합된 개발 환경과 에뮬레이터를 제공하여 iOS기반의 앱 시장을 넘어서고 있다. 통합된 개발 환경의 제공을 통해 기존의 C/C++, Objective-C, Java 개발자들이 쉽게 스마트폰 어플리케이션 개발자로 뛰어 들 수 있었다. 특히, 구글은 앞에서 본바와 같이 앱 인벤터를 통해 컴포넌트 기반의 어플리케이션 제작 도구를 제공하고 있다. 프로그래밍 언어를 알지 못하더라도 어플리케이션을 개발할 수 있도록 유도하고 있어 더욱 다양한 분야의 사람들이 더욱 다양한 콘텐츠를 제공할 수 있는 환경을 형성시키고 있다. 자동차 분야의 AUTOSAR 플랫폼도 이와 같은 맥락의 개발 환경을 제시하고 있다. 자동차 분야의 AUTOSAR 플랫폼을 이루고 서비스 계층의 컴포넌트들을 추상화시킨 API를 통해 어플리케이션을 개발할 수 있는 환경을 제공한다. 이로써 ECU에 실행되는 어플리케이션을 더욱 쉽게 구현할 수 있고 추상화된 API를 사용하기 때문에 하드웨어에 독립적으로 설계될 수 있어 하드웨어가 바뀌더라도 그대로 사용할 수 있는 장점이 있다. 로봇 소프트웨어를 재사용하기 위해서 로봇 소프트웨어 플랫폼이 발전하게 되었으며 OPRoS도 이와 같은 문제점을 해결하고자 개발되었다. OPRoS는 컴포넌트를 활용하여 개발의 편의성 및 재 활용성을 높였다. 또 OPRoS는 통합된 개발 환경을 제공하여 로봇 소프트웨어 전문가와 일반 개발자가 콘텐츠를 더 쉽게 개발하게 해준다. 그리고 OPRoS는 시뮬레이터까지 제공하여 로봇을 통해 직접 테스트하지 않아도 테스트 할 수 있는 환경을 제공하여 개발 비용과 시간을 더욱 단축시킬 수 있게 해준다.

OPRoS는 컴포넌트들의 연결로 구성되는데 컴포넌트 간의 점유 시간을 낮추기 위한 방법으로 포트(port)를 통한 컴포넌트 연결 방법을 적용하였다. 포트란 UML(unified markup language)2.0에서 추가된 개념으로 컴포넌트가 외부 환경과 상호 작용하는 지

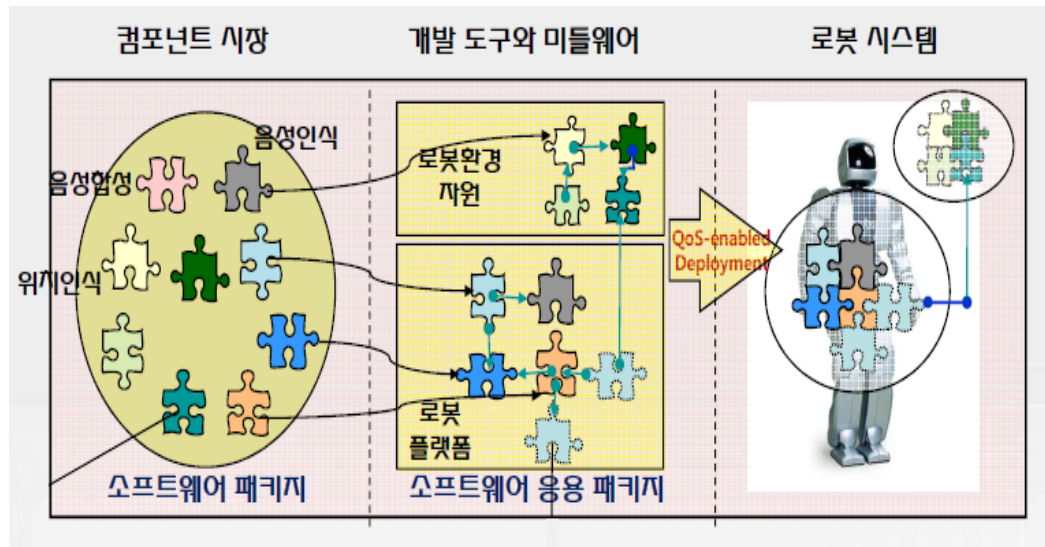
점을 의미한다. OPRoS 컴포넌트들은 포트를 통해 다른 컴포넌트들과 연결하고 상호 간 통신을 수행한다. 여기서 포트를 통해 연결하는 외부 컴포넌트 사용자는 컴포넌트가 제공하는 인터페이스만을 이용하며 해당 인터페이스에 대한 상세 구현은 알 필요가 없게 된다. 컴포넌트 사이의 통신은 송신 컴포넌트의 포트와 수신 컴포넌트 포트의 연결을 통해 이루어지는데 로봇 소프트웨어 컴포넌트 간 정보 교환은 주기적 데이터 전송과 이벤트 지원 및 원격 프로시저 호출이 있다. 이와 같은 정보 교환 방식을 지원하기 위해 OPRoS 컴포넌트에서는 각 교환 방식에 대응되는 세 가지 종류의 포트를 지원한다. 즉, 원격 메서드 호출을 위한 서비스 포트, 데이터 송수신을 위한 데이터 포트 및 이벤트 처리를 위한 이벤트 포트를 지원한다.



출처: OPRoS(<http://www.ropros.org>). 2016. 08. 04. 스크린샷.  
[그림 1-3] OPRoS 컴포넌트 모델

OPRoS 컴포넌트는 서비스 포트, 데이터 포트, 이벤트 포트 중 적어도 하나의 포트를 가지고 있어서 외부와의 인터페이스를 수행한다. 또 컴포넌트는 용도에 따라 다양하게 포트를 생성하여 사용할 수 있으며 같은 종류의 포트를 여러 개 생성하여 컴포넌트를 구성할 수 있다. 컴포넌트는 서비스 포트를 통해 다른 컴포넌트가 제공하는 메소드를 호출하거나 해당 컴포넌트의 속성에 접근할 수 있으며 데이터 및 이벤트 포트를 통해 해당 컴포넌트에 데이터나 이벤트를 전달할 수 있다. 컴포넌트 간의 메소드 호출이나 데이터/이벤트 전달은 모두 포트를 통해 이루어지기 때문에 다른 컴포넌트에 메소드를 호출하거나 데이터 및 이벤트를 전달하기 위해서는 해당 컴포넌트의 포트를 알아야

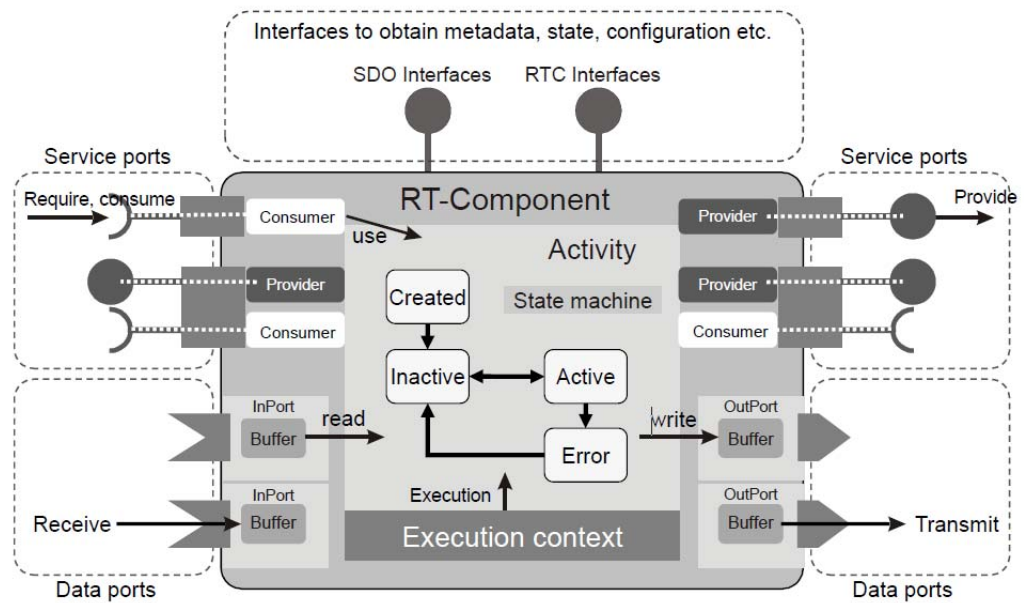
한다. 이를 위해 컴포넌트 개발자는 해당 컴포넌트가 제공하는 포트와 다른 컴포넌트가 사용할 수 있는 포트를 프로파일에 명시해야 한다. 서비스 포트는 컴포넌트 내부의 메소드들과 바인딩 되어 사용되며 데이터 포트는 데이터 처리기와 바인딩 되어 주기적인 방식이나 비주기적 방식으로 컴포넌트의 onExecute( ) 함수를 호출하여 데이터를 처리한다. 이벤트 포트는 이벤트가 도착 시에 특정 메소드(onEvent( ))가 즉시 호출되어 상태 관리를 하도록 한다.



출처: OPRoS(<http://www.ropros.org>). 2016. 08. 04. 스크린샷.  
[그림 1-4] OPRoS 컴포넌트 개념의 로봇 개발 과정

### (3) RTC(robot technology component)

RTC 규격은 로봇용 소프트웨어 컴포넌트들이 서로 조합되고 접속되기 위한 컴포넌트 사이의 공통 인터페이스 규격이다. 일본의 AIST와 미국의 RTI사는 RTC 표준 규격의 공동 개발 기관이다. RTC의 미들웨어 표준은 AIST가 개발하였고 관련 소프트웨어 미들웨어는 기업에서 사용하고 있기 때문에 새로운 표준 규격에 대한 구현이 어렵지는 않을 것으로 예상하고 있다. 특히, 로봇의 많은 소프트웨어 모듈이 내장형 소프트웨어 이므로 경량화한 컴포넌트 모델로 구현하도록 한 것이 RTC의 특징이다. 일본 측은 표준 규격의 구현을 위하여 일반적인 컴포넌트를 RTC 규격에 부합한 컴포넌트로 자동 변경하여 주는 툴을 개발하고 있다. 미국의 RTI사는 RTC를 이용한 미들웨어를 개발하여 이를 상용화하고 있으며 통신과 분산 환경을 제공하는 미들웨어 프레임워크를 개발할 것으로 보인다.



출처: 한국정보통신기술협회(<http://www.tta.or.kr>). 2016. 08. 04. 스크린샷.  
[그림 1-5] RTC 컴포넌트 구조

#### (4) 기타 로봇 운영 체제

iCub는 이탈리아의 IIT(istituto italiano di tecnologia)에서 개발한 휴머노이드 로봇의 운영 체제로 전 세계적으로 20개 이상의 연구소에 iCub를 채택하여 연구를 진행하고 있다. iCub는 전자회로도, 하드웨어 설계도, 소프트웨어 및 관련 자료까지 공개하였으며 운영 체제는 GPL(general public license) 기반으로 공개되고 있다. iCub 기반 로봇은 2~3세의 아이를 모티브로 개발이 진행되었으며 어린 아이가 인지를 통한 다른 사람과의 상호 작용을 통해 학습을 하는 것과 같이, 다양한 센서 시스템을 통해 아이와 같은 인지 능력을 가질 수 있다는 가설을 기반으로 개발되었다. 또 iCub는 다양한 국가의 공동 연구로 추진되어 지속적인 연구 개발이 가능한 환경으로 연구가 진행이 되고 있다는 특징을 가지고 있다.

OROCOS(open robot control software)는 EU와 벨기에(K. U. leuven), 프랑스(LAAS toulouse), 스웨덴(KTH stockholm)이 협력하여 진행하고 있는 실시간 제어 중심의 운영 체제 개발 프로젝트에서 시작되었다. 2001년 9월부터 본격적으로 개발을 시작해 2002년 여름에 첫 번째 버전이 발표되었다. OROCOS는 범용적인 무료 소프트웨어를 개발하고 로봇과 머신 제어를 위한 로봇 플랫폼에 독립적인 소프트웨어 프레임워크 개발을 목표로 하고 있다. OROCOS는 어플리케이션 개발을 위한 Real-Time Toolkit, Kinematics, Dynamics Library, Bayesian Filtering Library 와 같은 4개의 C++ 라이브러리를 지원하며 이전에 사용했던 컴포넌트 또는 타인에 의해 제공되는 컴포넌트들을 선택해 사용할 수 있는 재사용성의 장점이다. 그러나 OROCOS는 C++클래스 구조만 제공하며 개발 도구나 컴포넌트 등은 제공하지 않고 Real-Time Toolkit 위주의 컴포넌트

실행에 초점이 맞춰져 실시간 외의 서비스에 대해서는 적합하지 않은 구조를 갖고 있다. MRSRS(microsoft robotics studio)는 로봇 분야에 관심이 있는 사람들이 프로그래밍에 대한 기본적인 지식만 가지고 있으면 다양한 로봇 하드웨어 상에서 필요로 하는 다양한 로봇 어플리케이션을 개발할 수 있도록 지원하기 위한 목적으로 개발된 개발 툴 및 환경이다. 2006년 말에 출시되어 VPL(visual programming language)라는 tool을 제공하고 고급 개발자들의 경우 C#을 이용한 서비스나 시뮬레이션 프로그래밍을 할 수 있도록 지원하고 있다. MRSRS의 특징은 실제 로봇 하드웨어 없이 3D 물리엔진 기반의 시뮬레이션 환경을 이용해 개발된 프로그램을 테스트 할 수 있다는 점이다. 실제 로봇을 정교하게 제어하기 위해서는 다양한 물리적 요소를 감안한 프로그래밍이 필요하지만 실제 로봇 없이도 이를 가능하게 해 주는 물리엔진 기반의 시뮬레이터가 포함되어 있다.

이외에도 일본의 OpenRTM(AIST), NAOqi OS(Softbank), ERSP(Evolution Robotics) 등의 다양한 로봇 소프트웨어 운영 체제의 개발이 진행되고 있다. 아마도 로봇이 인간의 다양한 활동에 직접적인 영향을 미치는 시점에는 로봇 소프트웨어 운용 체제도 핸드폰 사례와 같이 통일화되어 높은 신뢰성과 안전성을 확보하는 시점이 올 것으로 예상된다. 물론, 로봇 미들웨어는 로봇 소프트웨어 운영 체제 안에 포함되어 다양한 기능이나 서비스를 연결하는 기능을 수행하게 될 것이다.

## 수행 내용 / 로봇 미들웨어 사양 분석하기

---

### 재료 · 자료

- 로봇에 대한 사용자 요구사항서
- 운영 체제 매뉴얼
- 로봇 미들웨어 사양서
- 로봇 미들웨어 설계서
- 로봇 미들웨어 프로그램 코드

### 기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 문서 작성용 소프트웨어

### 안전 · 유의 사항

- 로봇 소프트웨어 운영 체제를 분석하는 과정에 피학생자의 주관이 개입되지 않도록 조심하여야 한다.
- 로봇 소프트웨어 운영 체제 자료를 수집할 때 학습자들 간에 용어에 대한 오해가 생기지 않도록 로봇 소프트웨어에서 사용되는 용어의 정의에 대해 정확하게 공유하여야 한다.
- 학습자는 관련 자료를 수집함에 있어 수집 방법을 적절하게 배분하여야 한다.

### 수행 순서

#### ① 로봇 요구 사항에 대한 소프트웨어 운영 체제를 분석한다.

##### 1. 로봇 소프트웨어 운영 체제 기술 동향에 대해 조사한다.

(1) 로봇 소프트웨어 운영 체제에 대한 전반적인 흐름을 분석하여 대상으로 선정할 로봇 소프트웨어 운영 체제 결정한다. 수집 방법은 다음과 같은 것들이 있다.

(가) 인터넷을 통한 조사(naver, google 등 이용)

(나) 논문 조사([www.koreascience.or.kr](http://www.koreascience.or.kr), [scholar.google.co.kr](http://scholar.google.co.kr) 등과 같은 인터넷 이용)

(다) 한국정보통신기술협회: <http://www.tta.or.kr>

(라) 한국표준협회: <http://www.ksa.or.kr>

(마) 단체표준종합정보센터: <http://sps.kssn.net>

(바) RT미들웨어: <http://www.openrtm.org/openrtm/ko>

(사) JAVA: <http://www.oracle.com/technetwork/java/index.html>

(2) 다양한 자료가 수집되고 나면, 개발자들 간에 브레인스토밍 회의를 통하여 다양한 로봇 소프트웨어 운영 체제 중에서 로봇에 적용하기에 적합한 운영 체제를 선택한다.

## ② 로봇 소프트웨어 운영 체제에 대한 특징을 분류한다.

로봇 소프트웨어 운영 체제 중에서 로봇에 적용하기에 적합하다고 1차적으로 분류된 운영 체제를 대상으로 <표 1-2>과 같이 방법으로 미들웨어 구조 사양을 정리한다. 다양한 운영 체제들의 개발사부터 운영 체제를 선택하는 중요한 지표인 개발 tool, 오픈소스, 최적의 운영 체제 등에 대해서 조사를 자료를 바탕으로 작성한다. 특히, 구체적인 개발 과정에서 필요한 다양한 분류 항목에 대해서는 논의하여 추가하도록 한다.

<표 1-3> 로봇 소프트웨어 운영 체제 분류 예

번호	로봇 소프트웨어 운영 체제	개발사	개발 년도	최적 운영 체제	개발 Tool	오픈 소스	...
1	ROS	미국 (WG)	2010	Linux	○	○	
2	OPROS	한국	2009	Win/ Linux	○	○	
3	RTC	일본 (AIST)	2008	Win/ Linux	×	△	
4	iCub	유럽	-	-	○	△	
5	OROCOS	유럽	2002	Win/ Linux	×	○	
6	MRSRS	미국 (MS)	2006	Win/ Linux	○	△	
7	...						

## ③ 로봇 소프트웨어 운영 체제 및 미들웨어 역할 및 범위를 결정한다.

로봇의 기능과 소프트웨어 운영 체제 분석하여, 요구 사항 기능 정의서를 바탕으로 미들웨어 역할과 범위를 구체적으로 기술한다. 기 제공되는 미들웨어는 개발 시간의 단축과 효율을 높이는 데 중요한 역할을 한다. 또 로봇에 연결되어 있는 센서 및 액추에이터 운영 체제의 국제적인 흐름과 내용의 파악은 기본적으로 미들웨어 역할과 범위 정의에 중요하다 할 수 있다.



## 1. 미들웨어 역할 및 범위를 결정한다.

개발할 미들웨어의 역할과 범위를 로봇의 긴급 상황 발생 시 비상정지 및 운영 기능 정의를 기반으로 정의한다. 로봇 조작 시 발생하는 긴급 상황에서 로봇을 긴급 정지시키는 기능과 긴급 정지된 로봇을 대상으로 다시 동작시키는 기능으로 구분하여 미들웨어의 기능을 정의한다.

### (1) 로봇 긴급 정지 기능

(가) 로봇 긴급 정지 기능 제어기 입력 및 출력 신호

(나) 로봇 전원 제어기의 입력 및 출력 신호

(다) 로봇 긴급 정지 기능 동작 표시창 출력 신호

### (2) 로봇 긴급 정지 해지 기능

(가) 로봇 긴급 정지 해지 기능 제어기 입력 및 출력 신호

(나) 로봇 긴급 정지 해지 기능 배전반 입력 및 출력 신호

(다) 로봇 전원 제어기의 입력 및 출력 신호

(라) 로봇 긴급 정지 해지 기능 동작 표시창 출력 신호

## 수행 tip

- 국내에서 개발된 OPRoS를 대상으로 로봇 운영 체제 및 미들웨어 구현을 설명한다.
- 로봇 운영 체제와 미들웨어를 구분하여 설명하지 말고 운영 체제 내 하나의 기능으로 미들웨어를 설명하는 것이 적절하다.

## 2-1. 로봇 미들웨어 설계

### 학습 목표

- 로봇 요구 사항에 따르는 하드웨어를 구동하기 위한 통신 프로토콜을 설계할 수 있다.
- 로봇 요구 사항에 따르는 로봇 소프트웨어 공통 요소들을 호출하는 구조를 설계할 수 있다.
- 소프트웨어 공통 요소 및 응용프로그램들 간의 상호 정보 교환 및 프로세스 관리를 위한 미들웨어 구조를 설계할 수 있다.

### 필요 지식 /

#### ① 시스템 아키텍처(system architecture)

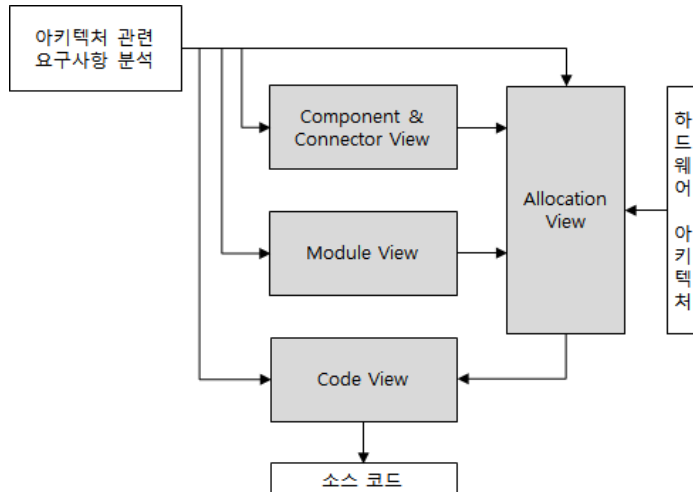
##### 1. 시스템 아키텍처 정의

시스템 아키텍처에 대한 정의는 다양하며 몇 가지 공통적인 사항을 정리하면 다음과 같다. 우선 시스템 아키텍처는 시스템의 구성 및 동작 원리를 도식화하여 표현한 것이다. 시스템의 구성 요소 혹은 구성 부품에 대한 설계 및 구현에 대한 상세한 사항을 기술한 것으로 정의하기도 한다. 시스템을 구성하는 다양한 구성 요소 간의 관계 및 시스템 외부 환경과의 관계가 상세하게 도식화되어 있으며 구체적인 구성 요소의 요구 사양 및 시스템의 전체 수명 주기를 기술하기도 한다. 마지막으로 시스템 전체(하드웨어와 소프트웨어를 포괄한 것)에 대한 논리적인 기능 체계와 그것을 실현하기 위한 구성 방식 및 시스템의 전체적인 최적화를 목표로 기술되어 있는 도식화된 기술서로 정의하기도 한다. 즉, 시스템이 어떻게 작동하는지를 설명하는 것으로, 구동 목적을 달성하기 위하여 시스템 각 요소가 무엇이며 어떻게 작용하여 정보를 교환하고 반응하는지를 기술로 정의할 수 있다.

##### 2. 소프트웨어 아키텍처 설계 절차

시스템 아키텍처 설계는 하위 구성 요소들 간의 관계를 정의하는 작업이다. 시스템 아키텍처 설계의 주요 목적은 시스템의 구조를 세분화하여 정의하고 하위 구성 요소들 간의

제어 및 정보 공유 관계를 기술하는 것이다. 시스템 아키텍처 설계를 위해 구조적 설계 기법에서는 자료 흐름도를 사용하고 객체지향 설계 기법에서는 클래스 다이어그램을 사용하여 점진적인 도식화 과정을 거친다.



[그림 2-1] 소프트웨어 아키텍처 설계 절차

## ② 소프트웨어 아키텍처 특성

설계 프로세스는 반복적인 활동들의 집합으로서 설계자는 이러한 반복 과정을 통해 개발할 소프트웨어 구조, 하부 단위, 정보, 인터페이스 등과 같은 소프트웨어의 모든 분야를 상세하게 설계하게 된다. 소프트웨어 아키텍처 설계는 개발할 소프트웨어의 전체 구조를 설계하는 기본적인 업무에서 하위 단위 사이에 주고받는 정보나 하위 단위에 대한 설계 같은 세부적인 사항을 포함한다.

### 1. 소프트웨어 아키텍처 추상화(abstraction)

일반적으로 복잡한 문제 해결을 위하여 초기에 세부 사항을 먼저 설계하기보다는 전체적이고 포괄적인 개념을 설계한 후 차례로 세분화하여 점진적으로 구체화시키는 방법이다. 단계적인 정제(refinement)를 통해서 기본 설계에서 상세 설계로 나아가면 추상화의 수준은 낮아지고, 원시 코드가 생성될 때 추상화는 최하위 수준이 된다. 설계 단계에서 사용되는 추상화의 예로는 제어 추상화, 과정 추상화 및 데이터 추상화가 있다.

<표 2-1> 소프트웨어 아키텍처 추상화 구분

번호	종류	내용
1	기능의 추상화	입력 자료를 출력 자료로 변환하는 과정을 추상화하는 방법
2	제어의 추상화	제어 절차를 정의하지 않고 원하는 효과를 정하는 데 이용하는 방법
3	자료의 추상화	자료와 자료에 적용될 수 있는 기능을 함께 정의함으로써 자료 개체를 구성하는 방법

## 2. 단계적 정제(stepwise refinement)

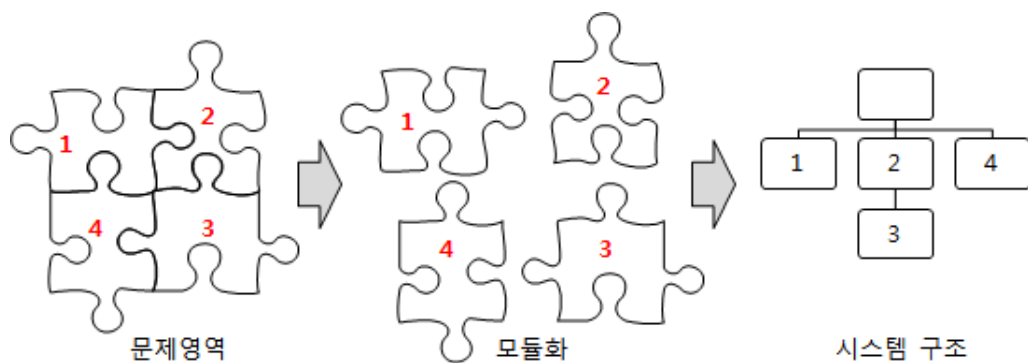
니클라우스 비르트(NIKLAUS WIRTH)에 의해 제안된 하향식 설계 전략이다. 단계적 정제는 문제를 상위의 중요 개념으로부터 하위의 개념으로 구체화시키는 분할 기법으로, 추상화의 반복에 의해 세분화된다. 소프트웨어 기능에서부터 시작하여 점차적으로 구체화하고, 상세한 내역(알고리즘, 자료 구조)은 가능한 한 뒤로 미루어 가면서 진행한다.

## 3. 정보 숨김(stepwise refinement)

정보 숨김은 필요하지 않은 정보는 접근할 수 없도록 제외하여 하나의 하위 구조 또는 하부 시스템의 자료, 절차에 대한 구체적인 구현 내용 및 내부 정보를 다른 외부 모듈의 접근으로부터 통제한다. 즉, 정의된 메시지에 의해서만 접근과 전달이 되도록 하는 방법으로 각 하위 구조 구현에 서로 영향을 받지 않게 설계되는 것을 의미한다. 소프트웨어 아키텍처 설계 단계에서 채택되는 설계 전략을 세부적으로 구분하여 설계 전략에 변경이 발생하는 경우, 그 영향이 최소한의 모듈들에만 미치도록 하는 것으로, 모듈 단위의 변경 및 시험이 쉬우며 유지보수 시 위험에 따른 영향이 최소화되는 큰 장점이 있다. 시스템 아키텍처 설계에서 숨겨야 할 기본 정보는 상세한 데이터 구조, 하드웨어 디바이스를 제어하는 부분, 특정한 환경에 의존하는 부분(특정한 운영 체제나 DBMS에 의존하는 부분) 및 물리적 코드(IP 주소, 문자 코드) 등이 있다.

## 4. 모듈화(modularity)

시스템 아키텍처 모듈화는 소프트웨어를 각 기능별로 분할하는 것을 의미하며 각 기능별로 분할한 것을 의미한다. 세부 하위 모듈은 서브루틴, 부(sub) 시스템, 소프트웨어 내의 프로그램, 작업 단위 등을 의미한다. 모듈화를 이용하여 설계하면 소프트웨어의 복잡도가 감소하고 변경이 쉬우며 프로그램 구현이 용이하고 확장성, 융통성, 경제성 등이 향상된다.



[그림 2-2] 소프트웨어 아키텍처 모듈화 사례

### (1) 모듈 속성

소프트웨어에서 모듈이란 한 프로그램의 세부적인 일부분으로 정의된다. 하나 또는 다

수의 논리적인 기능을 수행하기 위한 프로그램은 하나 이상의 독립적으로 개발된 모듈로 구성되며 이들은 그 프로그램이 접속되기 이전까지는 결합되지 않는다.

<표 2-2> 소프트웨어 모듈 속성

번호	외부 속성	내용
1	명칭	모듈이 다른 모듈을 호출할 때 사용
2	기능	그 모듈이 실행하는 기능은 간단한 문장으로 기술하고 모듈의 내부 논리는 제외
3	매개변수 목록	모듈과 다른 모듈과의 사이에서 주고받는 매개변수의 수나 순서 등을 명확히 정의
4	입력과 출력	매개 변수 목록상의 변수를 입력과 모듈의 출력으로 나누어 보다 상세히 기술
5	외부 효과	그 모듈이 실행하는 물리적인 입출력이 여기에 해당하며 실행되었을 경우, 프로그램이나 시스템 등 그 모듈이 외부 사항에 미치는 영향 및 효과를 기술

## (2) 모듈 결합도(coupling)

모듈 결합도는 모듈 간에 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계(정보 교환)를 의미한다. 모듈들 사이의 정보 교환이 많고 서로의 정보에 따라 구동되는 기능이 많을수록 모듈들 사이의 결합도는 높아진다. 독립적인 모듈이 되기 위해서는 각 모듈 간의 결합도가 약해야 하며 의존하는 모듈이 적어야 한다. 인터페이스가 정확히 설정되어 있지 않거나 기능이 정확히 세분화 되어있지 않을 때 불필요한 인터페이스가 나타나 모듈 간의 의존도와 결합도가 증가한다.

결합도는 모듈의 독립성 및 응집도와 밀접한 관계를 가지고 있다. 즉, 두 모듈이 완벽하게 기능을 수행한다면 이들은 서로 완전히 독립적이라고 할 수 있으며 서로 상호 교류가 전혀 없음을 의미한다. 결합도가 높을수록 한 모듈의 변화가 다른 모듈에도 영향을 주며 영향이 클수록 시스템을 유지보수하기 어려워진다.

### (가) 자료 결합도

자료 결합도는 모듈 간의 인터페이스가 자료 요소로만 구성된 경우의 결합도이다. 모듈이 다른 모듈을 호출하면서 데이터를 넘겨주고 호출 받은 모듈은 받은 데이터에 대한 결과를 다시 돌려주는 방식이다. 자료 결합도는 한 모듈의 내용을 변경하더라도 다른 모듈에는 전혀 영향을 주지 않는 결합 방식이다.

### (나) 스택프 결합도

스택프 결합도는 모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달

된 경우의 결합도이다. 모듈이 동일한 자료 구조를 조회하는 경우의 결합도이며 자료의 구조의 변화는 그것을 조회하는 모듈 및 조회하지 않는 모듈에까지도 영향을 미치는 결합 방식이다.

(다) 제어 결합도

제어 결합도는 한 모듈에서 다른 모듈로 논리적인 흐름을 제어하는 데 사용하는 제어 요소가 전달될 때의 결합도이다. 상위 모듈이 하위 모듈의 상세한 처리 절차를 알고 있어 이를 통제하는 경우나 처리 기능이 두 모듈에 분리되어 설계된 경우에 발생한다.

(라) 공유 결합도

공통 결합도는 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도이다. 공통 데이터 영역의 내용을 조금만 변경하더라도 이를 사용하는 모든 모듈에 영향을 미친다.

(마) 내용 결합도

내용 결합도는 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도이다. 한 모듈에서 다른 모듈의 중간으로 분기되는 경우에도 내용 결합도에 해당된다.

(3) 모듈 응집도(cohesion)

모듈 응집도는 정보 은닉 개념을 확장한 것으로 모듈 내부에서 처리 과정 및 정보의 생성, 소비가 이루어지는 정도를 의미한다. 모듈 내부 요소들 사이의 응집도가 증가하도록 하는 것이 바람직하며 모듈의 응집도를 높이면 모듈 사이의 낮은 결합도를 얻을 수 있다. 독립적인 모듈이 되기 위해서는 각 모듈의 응집도가 강해야 하며 응집도의 종류에는 기능적 응집도, 순차적 응집도, 교환(통신)적 응집도, 절차적 응집도, 시간적 응집도, 논리적 응집도, 우연적 응집도가 있다.

(4) 프로그램 구조(program structure)

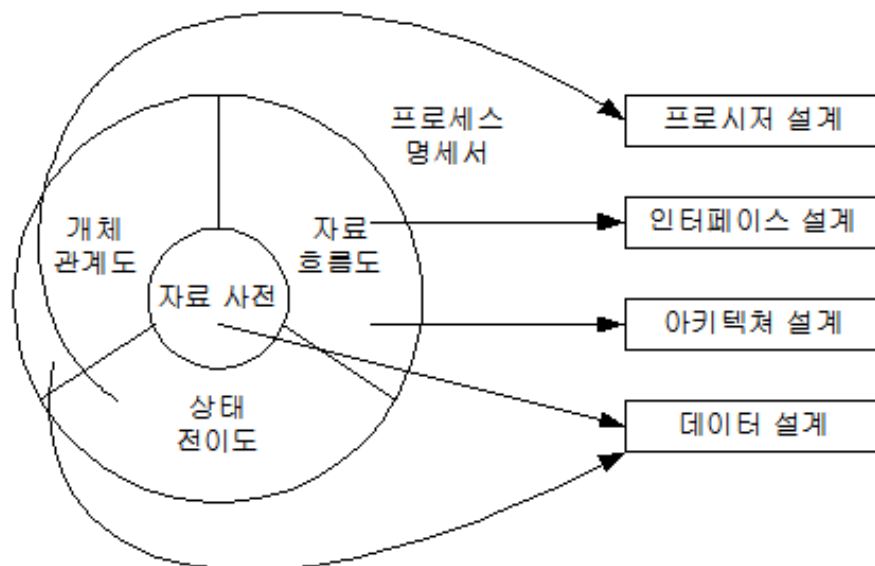
프로그램 구조는 소프트웨어를 구성하고 있는 모듈과 모듈 간의 상호 작용 및 모듈에 의해 사용되는 자료 구조를 의미한다. 프로그램 구조는 소프트웨어의 구조 요소인 모듈의 계층적 구성을 나타내는 것으로, 트리 구조의 다이어그램으로 표기한다.

<표 2-3> 프로그램 구조에 사용되는 요소

번호	종류	내용
1	공유도(fan-in)	얼마나 많은 모듈이 주어진 모듈을 호출(제어)하는가를 나타냄 주어진 모듈을 직접 불러 제어(호출)하는 상위 조정 모듈 수
2	제어도(fan-out)	어떤 모듈에 의해 호출(제어)되는 모듈의 수를 나타냄. 한 모듈이 직접 불러 제어(호출)하는 하위 계층 모듈의 수
3	깊이(depth)	제어 단계의 수
4	넓이(width)	제어의 전체적인 폭
5	주종적 모듈(superordinate)	다른 모듈을 제어(호출)하는 모듈
6	종속적 모듈(subordinate)	어떤 모듈에 의해 제어되는 모듈

### ③ 소프트웨어 아키텍처 설계 방법

소프트웨어 설계 모형을 기술적인 관점에서 분류하면 데이터 설계(data design), 구조 설계(architectural design), 인터페이스 설계(interface design), 절차 설계(processor design)로 구성된다.



[그림 2-3] 소프트웨어 아키텍처 설계 모형

<표 2-4> 소프트웨어 아키텍처 설계 모형

번호	종류	내용
1	데이터 설계	요구 사항 분석 단계에서 파악된 자료를 소프트웨어를 구현하는 데 필요한 자료구조로 변환하는 것으로 ERD(Entity Relationship Diagram)를 이용하여 정의된 개체와 관계, 자료 사전(Data Dictionary)에 정의된 자료의 구체적인 내용 등 설계
2	구조 설계	소프트웨어를 구성하는 모듈 간의 관계와 프로그램 구조를 정의하는 것으로 DFD(Data Flow Diagram), DD(Data Dictionary), STD (State Transition Diagram)등과 모듈의 상호 작용 설계
3	인터페이스 설계	소프트웨어와 상호 작용하는 시스템 사용자 통신 과정 설계
4	절차 설계	모듈이 수행해야할 기능을 절차적 기술한 것으로 소단의 명세서 및 상태 전이도 정보 설계

## 1. 객체지향 설계 방법

객체지향 설계 기법은 구조화 설계 기법의 단점을 해결하기 위해 개발되었다. 객체지향에서는 세상의 모든 사물을 프로세스와 데이터를 모두 갖는 객체(object)로 정의한다. 객체는 상태, 행동 그리고 식별성이라고 하는 3개의 성질을 갖고 있으며 실세계의 사상 가능성, 품질과 유연성의 향상, 그룹 개발의 가속화를 들 수 있다. 또 요구 분석, 설계, 프로그래밍의 공정을 순환적으로 반복하면서 소프트웨어를 개발할 수 있다.

### (1) UML(unified modeling language) 정의

UML은 소프트웨어 및 시스템 아키텍처를 모델링하는 데 사용되는 표준 비주얼 모델링 언어이다. UML이 객체 관리 그룹(object management group, OMG)의 표준이며 매우 융통성 있고 사용자 정의할 수 있도록 설계된 시각적인 언어이다. 업무 프로세스, 작업 흐름, 쿼리 순서, 애플리케이션, 데이터베이스, 시스템 아키텍처 등을 이해하기 위한 모델을 포함하여 많은 종류의 모델을 시각적으로 설계할 수 있도록 해준다.

UML은 프로젝트의 성공을 보장하는 것은 아니지만 많은 일들을 개선시킨다. 예를 들어 프로젝트나 조직을 바꿀 때에 새로 교육하는 비용을 감소시킨다. 또 여러 가지 도구나 공정, 영역 사이의 통합을 이끌어 낼 수 있다. 가장 중요한 것은, UML이 개발자들로 하여금 업무 가치를 생산하는 데 집중할 수 있게 하고, 그것을 성취할 수 있는 패러다임을 제공한다는 데 있다.



## (2) UML 다이어그램(diagram)

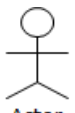
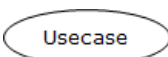
<표 2-5> UML 다이어그램 종류

번호	다이어그램명	내용
1	use case diagram	사용 사례와 사용자의 관계를 표현하는 다이어그램
2	행위적	sequence diagram 객체들 간의 상호작용을 시간적 순서를 강조하여 표현하는 다이어그램
3		collaboration diagram 객체들 간의 상호작용을 공간적 협조 체계를 강조하여 표현하는 다이어그램
4	statechart diagram	객체의 생명주기를 나타내며, 이벤트에 의해 변화하는 객체의 상태를 표현하는 다이어그램
5	activity diagram	객체에 작용하는 활동의 흐름을 표현하는 다이어그램
6	구조적	class diagram 클래스의 관계를 표현하는 다이어그램
7		component diagram 컴포넌트들의 관계를 표현하는 다이어그램
8		deployment diagram 시스템을 구성하는 물리적 객체나 장치들의 관계를 표현하는 다이어그램

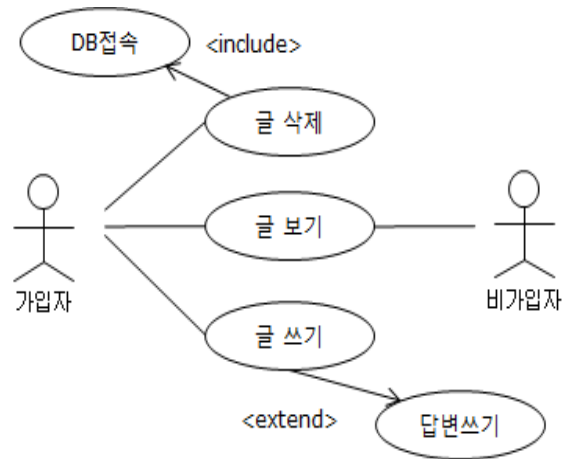
### (가) usecase diagram

누가 어떤 용도로 시스템을 사용하는지에 대해 행위자(actor)와 사용 사례(use case)의 관계로 표현할 수 있다.

<표 2-6> usecase diagram 구성 요소

번호	구성 요소	내용
1	 Actor	행위자(actor)는 시스템의 행위와 관련된 단위 작업별로 시스템을 사용하는 시스템 외부의 객체를 의미
2	 Usecase	쓰임새(use case)는 시스템이 행위자에게 제공하는 기능이나 서비스를 표현
3	통신	통신은 행위자와 쓰임새 간의 관계를 맺어주는 것으로 포함과 확장이 있으며 표시 할 때는 통신을 표시하는 선에 스테레오타입으로 명시

인터넷 사이트의 회원 게시판을 usecase diagram을 사용하면 다음과 같이 표현할 수 있다. 우선 비가입자는 글을 볼 수 있는 권한을 가지며 가입자는 읽고 쓰고 삭제하는 권한을 가진다. 또 가입자는 글쓰기에 답변쓰기 기능도 확장할 수 있다.



[그림 2-4] usecase diagram을 이용한 게시판 표현

#### (나) sequence diagram

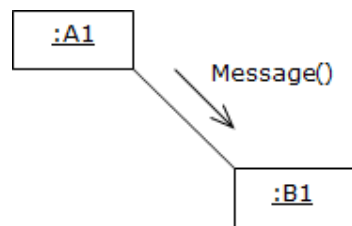
객체들의 행동을 시간의 관점에서 표현되며 객체에 실선 화살표로 그려지는 메시지, 그리고 수직 진행 상황을 나타내는 시간으로 구성된다. sequence diagram은 아주 단순하고 즉각적인 시각적 매력을 갖고 있기 때문에 자주 사용된다.

<표 2-7> sequence diagram 구성 요소

번호	구성 요소	내용
1		<p>각 객체로부터 아래로 뻗어가는 수직 점선은 객체의 생명선(lifeline)임. 생명선을 따른 사각형 모형을 실행(activation)이라 함. 객체가 수행하는 오퍼레이션이 실행되고 있음을 나타내며 실행 사각형의 길이는 오퍼레이션의 실행 소요 시간을 나타냄.</p>
2		<p>단순(simple) 메시지는 한 객체에서 다른 객체로 제어 흐름이 이동함을 의미.</p> <p>동기(synchronous) 메시지는 한 객체가 다른 객체로 보내는 메시지 객체로 수신 객체로부터 그 메시지를 받았다는 답신이 와야 다음 작업을 할 수 있음.</p> <p>비동기(asynchronous) 메시지는 동기 메시지와 달리 그 메시지를 전송한 후에 수신 객체로부터 답신을 기다리지 않고 다음 작업을 할 수 있음.</p>
3		<p>되부름(recursion)은 실행 사각형으로부터 작은 사각형으로 향하도록 화살표를 그린 다음 되부름을 시작한 객체 쪽으로 되돌아오는 화살표로 표현.</p>
4	[], *[]	if 문과 while 문을 표현하며 각각 [],*[]로 표현됨. 대괄호 안에 조건식을 삽입하면 됨.

#### (㉔) collaboration diagram

collaboration diagram은 교류를 수행하는 객체들의 전체적인 조직과 상황(context)에 초점을 맞춘 것이다. sequence diagram과 개념이 비슷하나 sequence diagram은 시간에 따라 배열되고, collaboration diagram은 공간에 따라 배열되어 있다. 두 객체 사이에 메시지는 두 객체 사이의 연관선과 가깝게 화살표를 그려준다. 화살표에는 번호가 붙는데, 이것은 메시지들의 순번을 나타내는 것이다. 또 오퍼레이션은 ‘메시지를 받는 객체로 하여금 어떤 오퍼레이션을 실행하라’는 뜻이다. 오퍼레이션 끝의 괄호 사이에 매개변수를 삽입할 수도 있다.

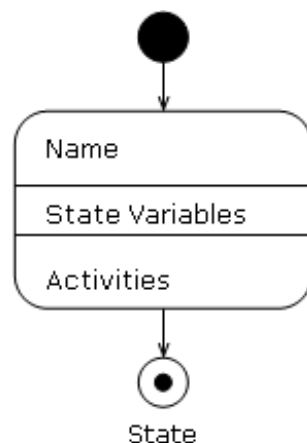


[그림 2-5] collaboration diagram 구성 요소

#### (㉕) statechart diagram

statechart diagram은 시스템의 변화 상태를 표현한다. 즉, statechart diagram은 하나의 객체에 집중하여 그 객체의 시작에서부터 종료까지의 상태변화를 표시하는 다이어그램이다. statechart diagram은 시스템의 모든 클래스에 대해 그릴 필요는 없으며 의미 있는 행동 양식을 보여주는 주요 클래스들에 대해서 그린다.

시작점(initial state)과 종료점(final state)이 있다. 이들은 각각 객체의 시작과 종료의 시점을 표시해 준다. 상태 다이어그램은 위에서부터 이름, 상태변수, 활동을 나타내는 세 영역으로 나눌 수 있다. activities는 사건(event)과 동작(action)으로 이루어진다.

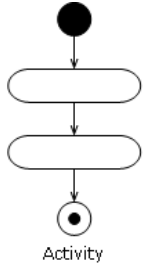




[그림 2-6] statechart diagram 구성 요소

(마) activity diagram

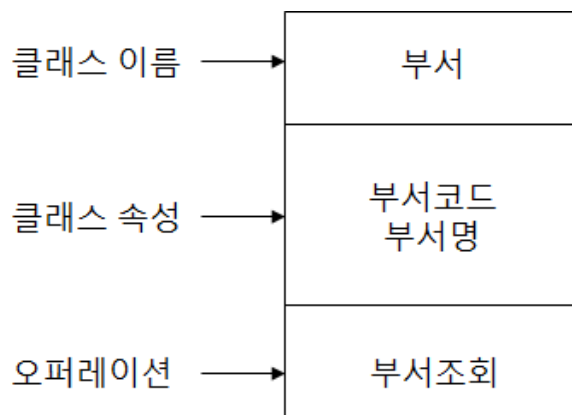
activity diagram은 오퍼레이션이나 처리 과정이 수행되는 동안 일어나는 일들을 간단하게 표현하기 위해 고안되었다. activity diagram은 처리 단계, 결정 위치, 분기 처리를 표현할 수 있어 업무 과정이나 오퍼레이션에서 처리되는 일 등을 효과적으로 나타내는 데 사용된다.

<표 2-8> activity diagram 구성 요소

번호	구성 요소	내용
1		시작점과 종료점이 존재하며 각각의 활동은 가늘고 모서리가 둥근 사각형으로 나타냄.
2		활동을 진행하다가 결정을 유하는 시점이 되면 마름모꼴을 사용하여 활동 전이를 나눔
3		두 가지 활동이 동시에 실행되는 것을 원할 경우 사용

(바) class diagram

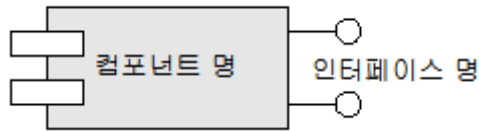
클래스(class)를 표현할 수 있는 양식으로 그 안에 속성(attribute)과 행위(operation)를 포함하고 있다.



[그림 2-7] class diagram 구성 요소

#### (사) component diagram

component diagram은 소프트웨어 컴포넌트의 구성과 연결 상태를 나타낸다. 소프트웨어 컴포넌트는 시스템을 이루는 물리적 요소이다. 그 예로서는 데이터 파일, 실행 파일, 도큐먼트 등이 있다. 여기서 컴포넌트는 ‘클래스를 소프트웨어로 구현한 것’으로 간단히 정의 내릴 수 있다.



[그림 2-8] component diagram 구성 요소

#### (아) deployment diagram

deployment diagram은 시스템의 소프트웨어와 하드웨어 컴포넌트 간의 물리적 관계를 표현한다.

## 수행 내용 / 로봇 미들웨어 설계하기

### 재료 · 자료

- 로봇 소프트웨어 운영 체제 분석 자료
- 로봇 기능 정의서
- 로봇 요구 사항 정의서
- 미들웨어 역할 및 범의 정의서
- UX/UI 기능 정의서

### 기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 문서 작성용 소프트웨어

### 안전 · 유의 사항

- 로봇 미들웨어 구조를 분석하는 과정에 피학생자의 주관이 개입되지 않도록 조심하여야 한다.

- 로봇 미들웨어 구조 설계 자료를 수집할 때 학습자들 간에 용어에 대한 오해가 생기지 않도록 로봇 미들웨어 구조 설계에서 사용되는 용어의 정의에 대해 정확하게 공유하여야 한다.
- 학습자는 관련 자료를 수집함에 있어 수집 방법을 적절하게 배분하여야 한다.

## 수행 순서

① 로봇 기능 중에서 긴급 정지 및 해지 기능에 대해서 토론을 통하여 분석한다.

1. 긴급 정지 및 해지 기능이 필요한 경우를 토론한다.

- (1) 로봇의 긴급 정지 기능은 작업 중 로봇을 정지시켜야 하는 위험 상황에서 동작하는 기능을 의미한다.
- (2) 로봇 긴급 정지 기능은 제어 패널 내 비상정지 버튼을 누름으로 긴급하게 로봇을 정지시킬 수 있다.
- (3) 로봇 긴급 정지 기능을 해지하는 경우 제어 패널 내 비상정지 버튼을 회전시키고 servo on ready 버튼을 누름으로 해지되는 경우와 조작반 내 비상정지 버튼을 회전시키고 데드맨 스위치를 누름으로 해지는 경우가 있다.

2. 긴급 정지 및 해지 기능이 필요한 경우에 대한 토론 결과를 통해 로봇 기능 정의서를 바탕으로 긴급 정지 및 해지 기능 절차를 토론하여 분석한다.

- (1) 로봇이 동작 상태에서 비상정지 버튼이 눌러지면 정지하게 된다.
- (2) 로봇이 정지된 상태에서 다시 동작을 하기 위해서는 비상정지 버튼을 회전을 시켜야 한다.
- (3) 비상 정지 버튼이 회전하면 회전이 된 비상정지 버튼이 패널에 있는 것인지 조작반에 있는 것이 확인을 위하여 패널에 있는 패널 버튼이 눌러졌는지 확인한다.
- (4) 패널의 버튼이 눌러졌다면 패널의 비상정지 버튼을 눌러졌다고 판단하고 서보 온 레디 버튼이 눌러졌는지 확인한다.
- (5) 서보 온 레디 버튼이 눌러졌다면 다시 정상적으로 동작을 하면 되고 눌러지지 않았다면 서보 온 레디 버튼이 눌러질 때까지 반복적으로 확인을 하며 대기한다.
- (6) 패널의 버튼이 눌러지지 않았다면 조작반의 비상정지 버튼이 눌러졌다고 판단하고 데드맨 스위치가 눌러졌는지 확인한다.
- (7) 데드맨 스위치가 눌러졌다면 다시 정상적으로 동작을 하면 되고 눌러지지 않았다면 데드맨 스위치가 눌러질 때까지 반복적으로 확인을 하며 대기한다.

- ② 로봇 기능 중에서 긴급 정지 및 해지 기능에 대해서 미들웨어 아키텍처를 설계하기 위하여 적합한 UML 다이어그램을 선정한다.

<표 2-9> UML 다이어그램 적합도 분석

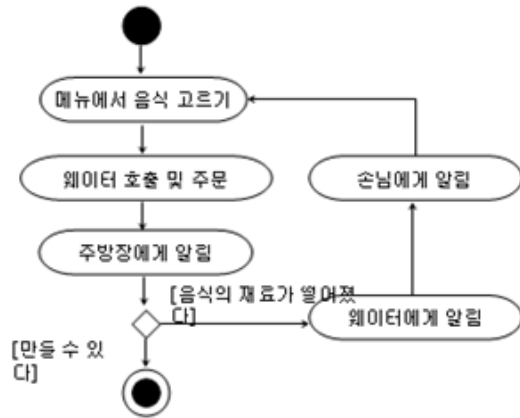
번호	다이어그램명	적합도			특징
		상	중	하	
1	use case diagram			○	일생 업무 표현에 적합
2	sequence diagram	○			순차적인 절차 표현에 적합
3	collaboration diagram		○		상호 관계 표현에 적합
4	statechart diagram		○		상태 기반 표현에 적합
5	activity diagram	○			과정 설계에 용이하면 표현이 쉬움
6	class diagram			○	그룹 표현에 용이
7	component diagram		○		하나의 그룹으로 쉽게 표현할 수 있음
8	deployment diagram			○	관계 표현 방법이나 복잡하고 어려움

1. UML 다이어그램을 대상으로 로봇의 긴급 정지 및 해지 기능의 아키텍처를 설계하기에 적합성을 분석한다.

UML 다이어그램 중에서 동작이나 처리 과정이 수행되는 동안 일어나는 일들을 간단하게 표현할 수 있는 activity 다이어그램으로 긴급 정지 및 해지 기능의 아키텍처를 설계한다. 긴급 정지 및 해지 기능에 대한 구체적인 동작 방법은 UX/UI 개발 지침서를 참고하여 긴급 정지 동작과 해지 동작을 구분한다. 또 제어 패널과 배전반에서의 긴급 정지 기능 해지에 대해서도 구분하여 설계를 진행한다.

2. 선정된 UML 다이어그램에 대한 추가 자료 조사 및 다이어그램 작성 방법을 함께 토론하여 작성할 수 있는 기초 지식을 습득한다.

activity 다이어그램 작성 사례를 연습하여 로봇 긴급 정지 및 해지 기능을 작성할 수 있는 지식을 습득한다. 예제는 중국집에서 음식을 주문하는 과정을 activity 다이어그램으로 작성한 사례이다. 우선, 손님이 중국집에 들어오면 메뉴판을 보고 음식을 선택한다. 음식을 선택한 손님은 웨이터를 불러서 주문을 한다. 웨이터는 손님이 주문한 음식을 주방에 알린다. 여기서 음식 재료가 떨어졌다면 주방에서 웨이터에게 음식 재료가 없다고 알린다. 웨이터는 이 사실을 손님에게 알리고 처음 단계로 돌아가는 과정이다.



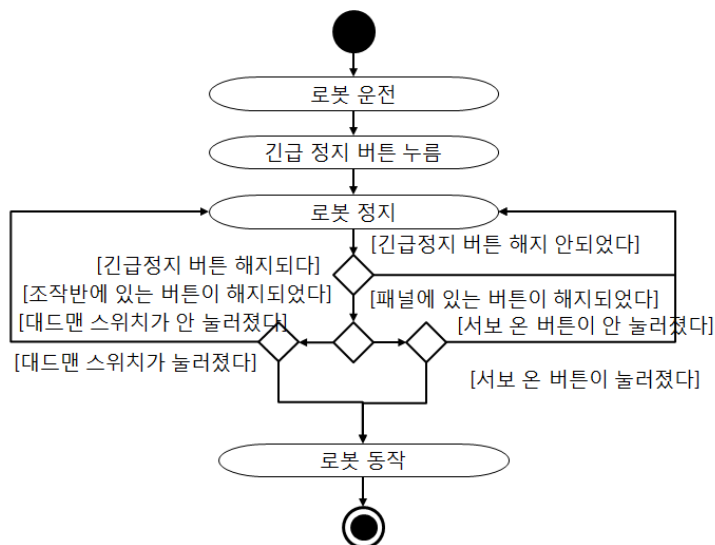
[그림 2-9] 중국집에서 음식을 시키는 과정을 activity diagram으로 설계한 사례

③ 로봇 기능 중에서 긴급 정지 및 해지 기능에 대해서 미들웨어 아키텍처를 설계한다.

1. 긴급 정지 및 해지 기능을 아키텍처로 설계할 때 필요한 활동 상태를 분류한다.

- (1) 비상정지 버튼 누름
- (2) 다시 동작 시 비상정지 버튼을 회전
- (3) 패널 및 조작반에 있는 것이 확인
- (4) 패널의 경우 서보 온 레디 버튼 확인
- (5) 조작반의 경우 데드맨 스위치 확인
- (6) 비상정지 기능 해지

2. 긴급 정지 및 해지 기능을 아키텍처 설계한다.



[그림 2-10] 긴급 정지 및 해지 기능을 activity diagram 설계한 사례



#### 수행 tip

- 미들웨어 아키텍처 설계 보다는 전반적인 소프트웨어 구조 설계를 중심으로 지도하는 것이 효율적이다.
- 통신 프로토콜은 제공하는 운영 체제에서 기본적으로 설계되어 있어 별도로 설계하는 경우가 없다.

학습 1	로봇 미들웨어 사양 분석하기
학습 2	로봇 미들웨어 설계하기
<b>학습 3</b>	<b>로봇 미들웨어 구현하기</b>

## 3-1. 로봇 미들웨어 구현

### 학습 목표

- 로봇 요구 사항에 따르는 하드웨어 통신 프로토콜 프로그램을 작성할 수 있다.
- 로봇 요구 사항에 따르는 로봇 소프트웨어 공통 요소들의 호출 구조를 API로 작성할 수 있다.
- 소프트웨어 공통 요소 및 응용프로그램들 간의 상호 정보 교환 및 프로세스 관리를 위한 미들웨어 프로그램을 작성할 수 있다.
- 특정 요구 기능과 독립하여 소프트웨어 공통 요소 및 응용프로그램의 예시 프로그램을 작성하여 상호 정보 교환 및 프로세서 관리의 동작을 확인할 수 있다.

## 필요 지식 /

### ① 로봇 소프트웨어 계층 구조

#### 1. 임베디드 소프트웨어 계층 구조

로봇 미들웨어 구현을 위한 첫 단계는 로봇 소프트웨어의 계층 구조를 이해하는 것이다. 여기서 소프트웨어 계층 구조는 요구 사항을 만족하는 시스템을 구축하기 위해서 전체 시스템에 대한 구조를 정의하는 것이다. 또 시스템을 구성하는 컴포넌트와 그 컴포넌트 간의 관계 그리고 컴포넌트가 다루는 정보를 정의하는 것이다. 소프트웨어 계층 구조는 현재의 요구 사항뿐만 아니라 변화되는 전략에 대응 가능하도록 장기적인 로드맵을 수용하여 확장 가능한 형태로 디자인되어야 한다. 또 구현 및 사용하고자 하는 조직의 기술 수준, 조직의 규모와 형태에 맞춰서 설계되어야 한다. 소프트웨어 계층 구조는 계층 구조를 구성하는 요소의 일부가 변경되거나 기존의 계층 구조에 새로운 요소가 추가되는 경우, 다른 요소들이 받는 영향이 최소화될 수 있도록 계층 구조는 유연(flexible)할 필요가 있다. 따라서 계층 구조 설계 프로세스의 핵심은 시스템의 공통성(commonalities)과 가변성(variabilities)을 파악하여 계층 구조를 정의하는 것이다.

어플리케이션(GUI)
미들웨어
디바이스 드라이버
운영체제
하드웨어

[그림 3-1] 임베디드 소프트웨어 계층 구조

#### (1) 어플리케이션(graphic user interface)

어플리케이션은 넓은 의미에서 운영 체제에서 실행되는 모든 소프트웨어를 의미한다. 예로 우리가 사용하는 PC에서 사용하는 워드프로세서, 한글, 웹브라우저뿐만 아니라 컴파일러나 링커 등도 응용 소프트웨어인 셈이다. 또 좁은 의미에서는 운영 체제 위에서 사용자가 직접 사용하게 되는 소프트웨어를 의미한다. 이런 경우 컴파일러나 링커 등 시스템 소프트웨어를 제외한 워드프로세서 등의 소프트웨어들을 주로 뜻한다. 이렇게 뜻을 한정할 경우 응용 소프트웨어는 시스템 소프트웨어의 여집합이라고도 생각할 수 있다. 간단하게 줄여서 애플리케이션이라고 하며 더 줄여서 핸드폰에서 사용하는 카카오와 같은 앱(app)이라고 부르기도 한다.

#### (2) 디바이스 드라이버

디바이스 드라이버는 특정 하드웨어나 장치를 제어하기 위한 커널의 일부분으로 동작하는 프로그램이다. 컴퓨터를 구성하는 다양한 입출력 장치마다 각각 디바이스 드라이버가 프로그램 되어 커널에 통합되어 실행된다. 참고로 커널은 운영 체제의 핵심적인 부분으로 하드웨어와 프로세서의 보안을 책임지며 시스템 자원을 효율적으로 관리하여 프로그램의 실행을 원활하게 하는 역할을 하는 소프트웨어이다. 높은 수준의 컴퓨터 프로그램들이 컴퓨터 하드웨어 장치와 상호 작용하기 위해 만들어진 하나의 컴퓨터 프로그램이다. 디바이스 드라이버는 커널의 일부분이기는 하나 커널과 통합되는 것은 처음부터 해당 드라이버 프로그램 코드가 커널 전체 코드에 포함되어 컴파일되는 경우도 있고, 별도로 컴파일된 파일(윈도의 \*.sys, 리눅스의 \*.o)의 형태로 존재하고 부팅 시 또는 필요 시 해당 파일이 로드되어 커널과 통합되기도 한다.

보통 소형 임베디드 시스템의 경우 OS를 설치하지 않는 경우가 많기 때문에 디바이스 드라이버가 존재하지 않을 수 있다. 이럴 경우 개발자가 프로그램을 통해 하드웨어를 직접 제어하는 것이다. 중대형 임베디드 시스템의 경우 운영 체제를 설치하면 필수적으로 디바이스 드라이버가 있는데 이는 직접 하드웨어를 제어하는 것이 아니라 운영 체제를 걸쳐 디바이스 드라이버를 통해 하드웨어를 제어하는 것이다.

### (3) 운영 체제

운영 체제는 시스템 하드웨어를 관리할 뿐 아니라 응용 소프트웨어를 실행하기 위하여 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어이다. 기본적으로 입출력과 메모리 할당과 같은 하드웨어 기능의 경우 운영 체제는 응용프로그램과 컴퓨터 하드웨어 사이의 중재 역할을 수행한다. 그러나 응용프로그램 코드는 일반적으로 하드웨어에서 직접 실행된다. 최근의 운영 체제는 휴대 전화, 게임기에서부터 슈퍼컴퓨터, 웹 서버에 이르기까지 컴퓨터를 포함하는 거의 모든 장치에서 볼 수 있다. 운영 체제는 소비자와 프로그램 개발자를 함께 하나의 시장으로 데려다 놓을 수 있는 양면 플랫폼이며 현대의 PC 운영 체제에는 마이크로소프트 윈도우, 맥 OS X, 리눅스가 있다.

### (4) 임베디드(embedded) 소프트웨어

임베디드 시스템은 기계나 기타 제어가 필요한 시스템을 대상으로 제어를 위한 특정 기능을 수행하는 컴퓨터로서 장치 내에 존재하는 전자 시스템이다. 임베디드 시스템은 전체 장치의 일부분으로 구성되며 제어가 필요한 시스템을 위한 핵심적인 두뇌 역할을 하는 특정 목적의 컴퓨터 시스템을 의미한다. 반면, 개인용 컴퓨터와 같은 특정되지 않는 일반적인 목적을 수행하는 컴퓨터 시스템과는 대조된다. 특정 목적을 수행하는 컴퓨터 시스템이므로 목적을 설정하고 이를 수행하는 프로그램 코드를 작성하여 메모리에 기록하고 이를 읽어 동작시키는 방법이 일반적이다. 전자 하드웨어와 기계 부분을 포함하는 전체 장치의 일부로 내장되었다는 의미에서 임베디드라는 단어로 사용되었으며 오늘날 일상생활에 쓰이는 많은 장치들이 임베디드 시스템으로 제어되고 있다. 임베디드 시스템에서 핵심적인 두뇌 역할은 마이크로 컨트롤러(microcontroller)나 DSP(digital signal processor) 등의 프로세서 코어가 담당하고 있으며, 많은 감지기가 상태 정보를 제공하고 구동기가 마이크로 컨트롤러의 명령에 따라서 동작하고 있다. 이와 같은 임베디드 시스템을 운영하는 소프트웨어를 임베디드 소프트웨어라고 하며, 로봇 시스템은 임베디드 시스템을 기반으로 개발되고 있다.

### (4) 펌웨어(firmware)

펌웨어는 장치 안에 있는 ROM 안에 저장된 소프트웨어를 말한다. 메인보드에 탑재된 바이오스(BIOS) 프로그램도 펌웨어라 볼 수 있다. 임베디드의 경우 사용자가 직접 프로그램을 플래시 메모리나 EEPROM에 적재하여 하드웨어를 동작시키는 경우가 있는데 이 프로그램이 영구적으로 저장되기 때문에 펌웨어라 부르기도 한다.

### (5) 미들웨어

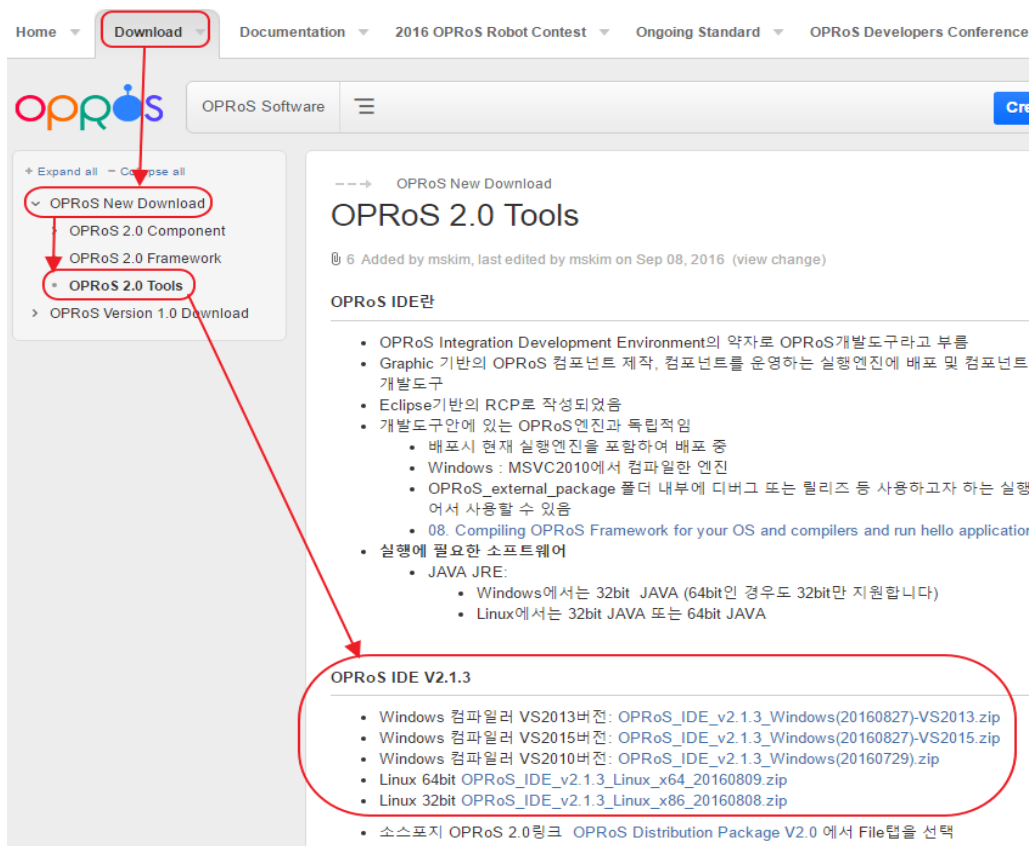
미들웨어는 애플리케이션에 도움이 되는 기능을 운영 체제나 하드웨어 등을 은폐한 형태로 제공하는 소프트웨어를 말한다. 임베디드 시스템의 경우에는 임베디드 운영 체제의 기능 제공 수준에 의하여 미들웨어의 기능 범위가 달라진다.

## ② 미들웨어 구현을 위한 운영 체제 설치

일반적으로 미들웨어는 임베디드 운영 체제 프로그램에 함께 설치되어 공통적으로 공유하는 정보를 설정하는 방법으로 구현한다. 특히, 로봇의 경우 다양한 운영 체제가 개발되어 경쟁적으로 표준이 되고자 노력하고 있다. 하지만 지역적인 문제와 경제적인 문제로 인하여 각 국가에서 개발한 로봇 운영 체제가 범용 표준으로 될 가능성이 큰 것으로 예상된다. 우리나라의 경우도 국가 주도로 로봇 전용 OPRoS(open platform for robotic service) 운영 체제를 개발하여 보급하고 있다. OPRoS는 컴포넌트 기반의 오픈소스 플랫폼이며 통합 개발 도구와 로봇에서 동작하는 프레임워크, 서버, 시험 및 검증 도구로 구성되어 있다. OPRoS에 대한 구체적인 정보는 공식 홈페이지([www.ropros.org](http://www.ropros.org))를 통하여 얻을 수 있다.

### 1. OPRoS 개발도구 설치

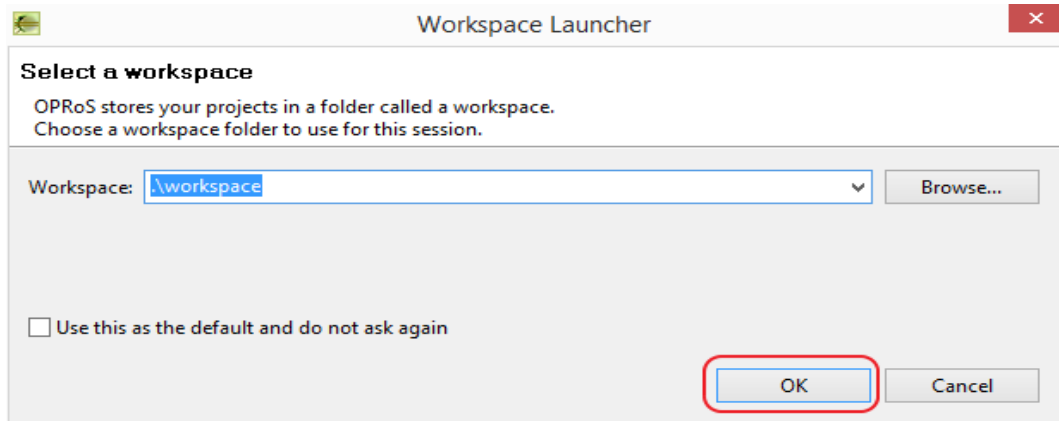
일반 PC 환경(Window7/8 32bit) 기준 MICROSOFT사의 visual Studio 2010/2013/2015를 설치한 환경에서는 공식 홈페이지([www.ropros.org](http://www.ropros.org))에서 그림 3-2와 같이 개발도구를 다운로드 받아 설치한다. 사용자는 사용하는 Visual Studio 버전에 맞는 OPRoS IDE 최신 버전을 다운 받고 압축을 푼다.



[그림 3-2] OPRoS 프로그램 다운로드 방법

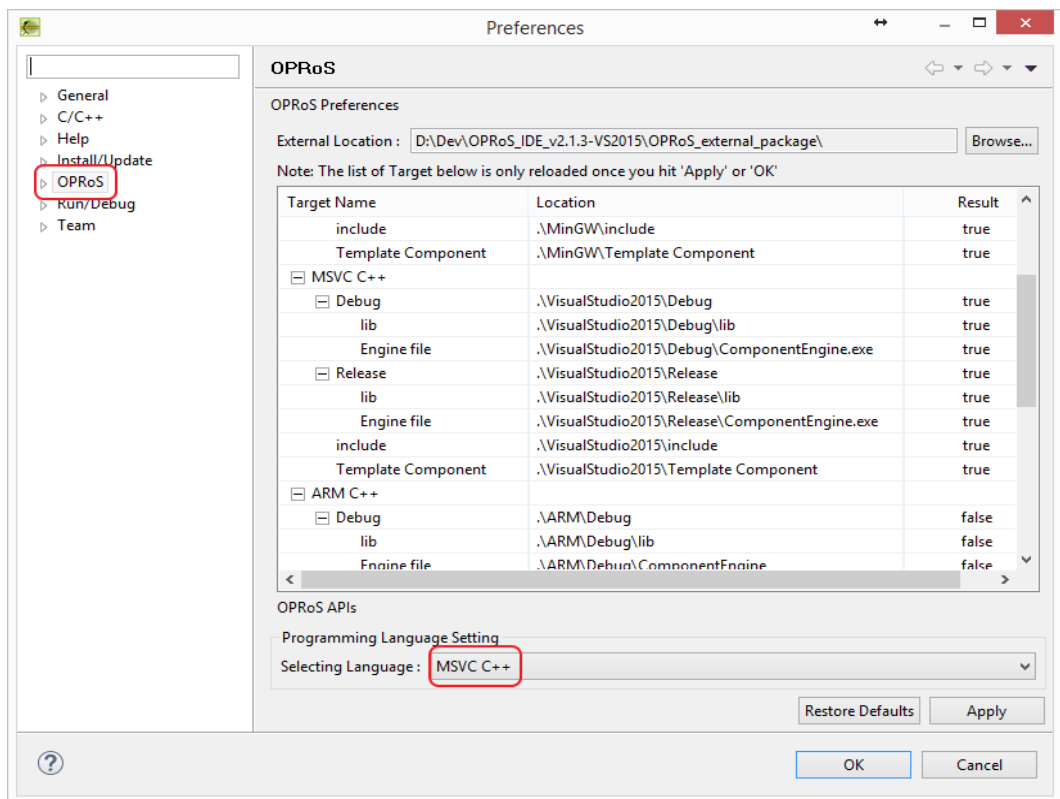
## 2. 환경 설정

다운로드 받아 압축을 푼 실행 프로그램에서 OPRoS\_IDE.exe를 실행하여 OPRoS 개발도구를 설치한다. 설치 경로는 별도로 설정하지 않고 기본 설정으로 되어 있는 Workspace 경로로 유지하고 OK 버튼을 눌러서 그림 3-3과 같이 프로그램을 설치한다.



[그림 3-3] OPRoS 프로그램 workspace 설정

OPRoS 개발도구를 실행한 뒤 상단 메뉴에서 Window/Preferences를 선택하고 이후 나타나는 Preferences 창의 왼쪽 메뉴에서 OPRoS를 선택하고 OK 버튼을 눌러 계속 설치를 진행한다.

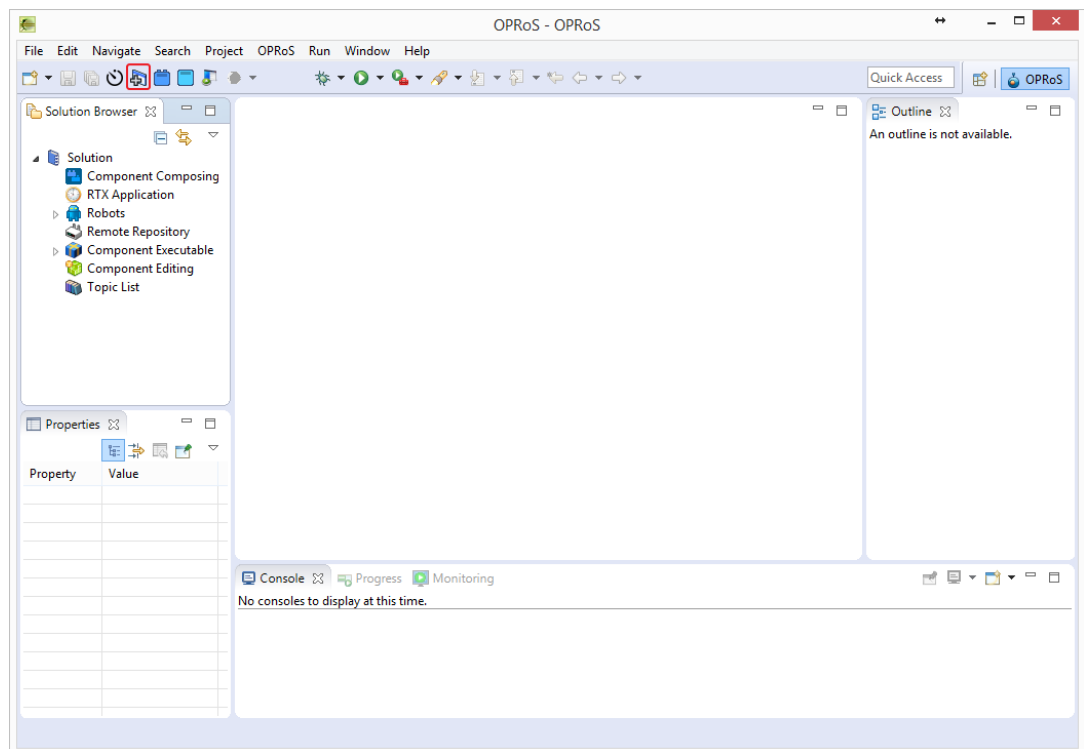


[그림 3-4] OPRoS 프로그램 preferences 설정

여기서, 하단의 Programing Language Setting 부분의 설정은 본인이 사용하고자 하는 프로그램을 선정하면 된다. Visual Studio를 사용하려면 MSVC++로 설정해야 하며 기타 프로그램을 사용할 경우 해당되는 프로그램으로 설정을 변경해야 한다. 여기서 설정 변경을 하지 않은 경우 컴포넌트 빌드 시 컴파일러를 찾지 못하는 에러가 발생할 수 있다.

### 3. 어플리케이션 생성

OPRoS 프로그램을 다운 받아 설치가 완료되면 신규 어플리케이션을 하나 생성하는 작업을 진행한다. 여기서 어플리케이션은 하나의 프로그램을 이식할 수 있는 공간으로 이해하면 쉬우며, 한글 프로그램에서 하나의 새로운 창을 열어서 새로운 글쓰기 창을 만드는 것과 같이 이해하면 된다. 메뉴에서 File> New> New Application을 선택하면 그림 3-5와 같은 화면이 생성된다.



[그림 3-5] OPRoS 프로그램 어플리케이션 생성 화면

## 수행 내용 / 로봇 미들웨어 구현하기

---

### 재료 · 자료

- 로봇 소프트웨어 운영 체제 분석 자료
- 로봇 기능 정의서
- 로봇 요구 사항 정의서
- 미들웨어 역할 및 범의 정의서
- UX/UI 기능 정의서
- 로봇 긴급 정지 및 해지 기능 activity diagram

### 기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 문서 작성용 소프트웨어

### 안전 · 유의 사항

- 로봇 미들웨어 구현에 과정에 피학생자의 주관이 개입되지 않도록 주의하여야 한다.
- 개별 학습자는 각자 로봇 미들웨어를 포함하는 로봇 소프트웨어를 직접 구현하고 실행시켜 실습의 효과를 최대한 확대시킬 수 있도록 실습을 수행해야 한다.

### 수행 순서

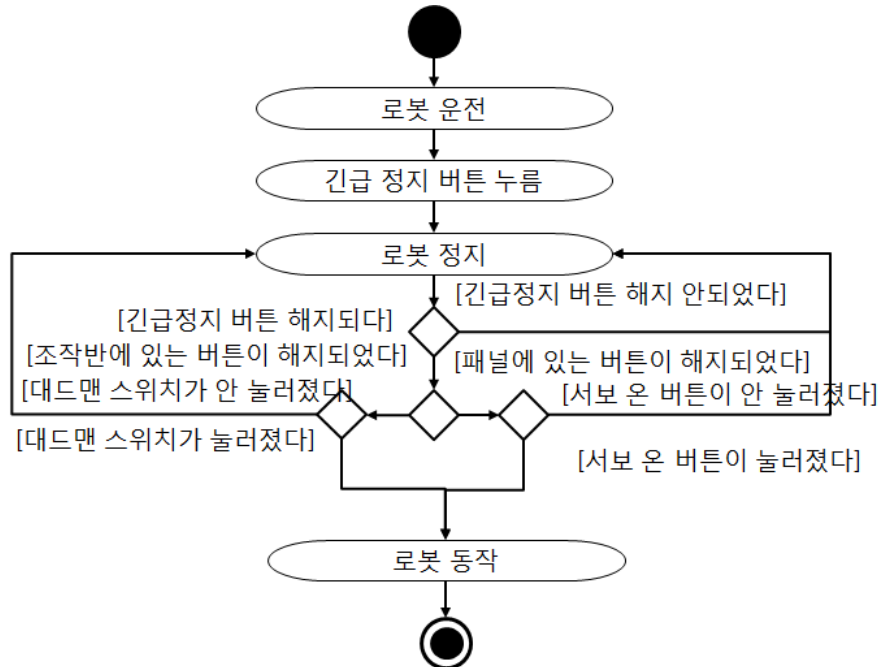
① 로봇 기능 중에서 긴급 정지 및 해지 기능의 activity diagram에 대해서 토론을 진행하여 긴급 정지 및 해지 기능에 대해서 상세하게 분석한다.

1. 긴급 정지 및 해지 기능 구조를 분류한다.

- (1) 비상정지 버튼 누름
- (2) 다시 동작 시 비상정지 버튼을 회전
- (3) 패널 및 조작반에 있는 것이 확인
- (4) 패널의 경우 서보 온 레디 버튼 확인
- (5) 조작반의 경우 데드맨 스위치 확인
- (6) 비상정지 기능 해지



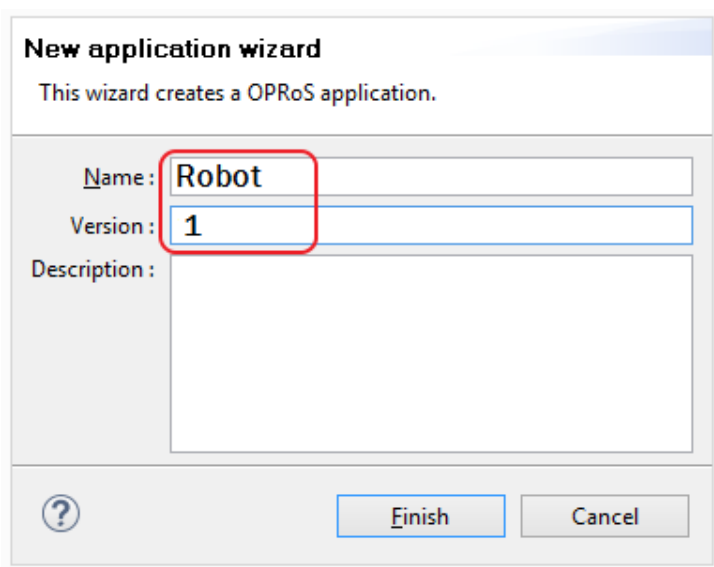
2. 설계한 긴급 정지 및 해지 기능 아키텍처를 점검하고 어떻게 소프트웨어로 구현할지 토론한다.



[그림 3-6] 긴급 정지 및 해지 기능 구조 설계

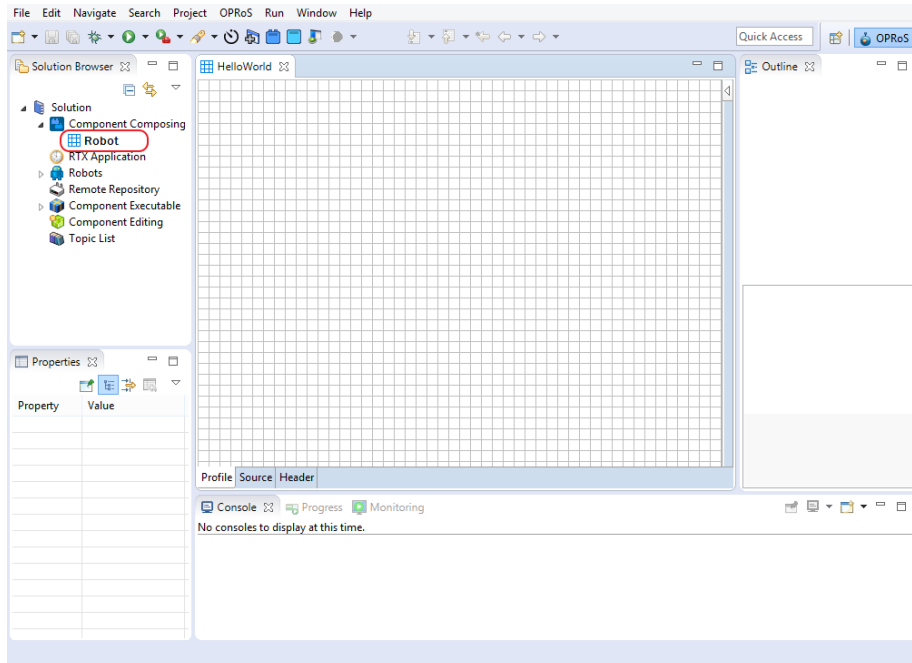
- ② 긴급 정지 및 해지 기능을 OPRoS를 이용하여 구현한다.

1. OPRoS에서 File > New > New Application을 설정하여 어플리케이션을 생성한다.
2. New Application Wizard가 실행되면 Robot이라는 이름과 버전 [그림 3-7]과 같이 설정한다.



[그림 3-7] New Application Wizard 설정

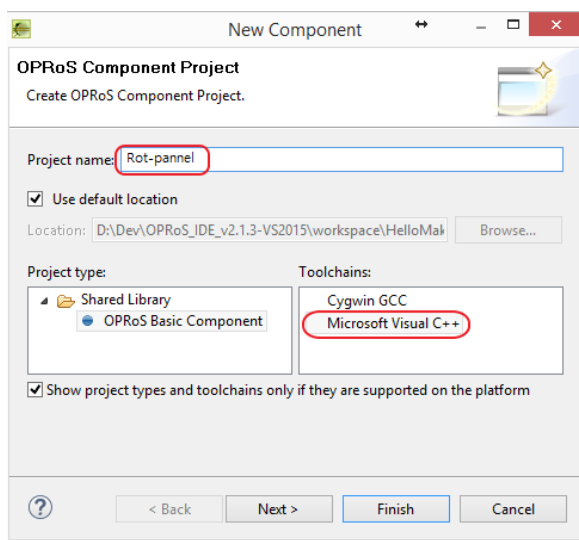
3. [그림 3-8]과 같이 새로운 어플리케이션이 생성되며 Solution Browser의 Component Composing에 Robot이 생성되고 가운데 창에 모눈이 생성된다.



[그림 3-8] 새로운 어플리케이션 생성 화면

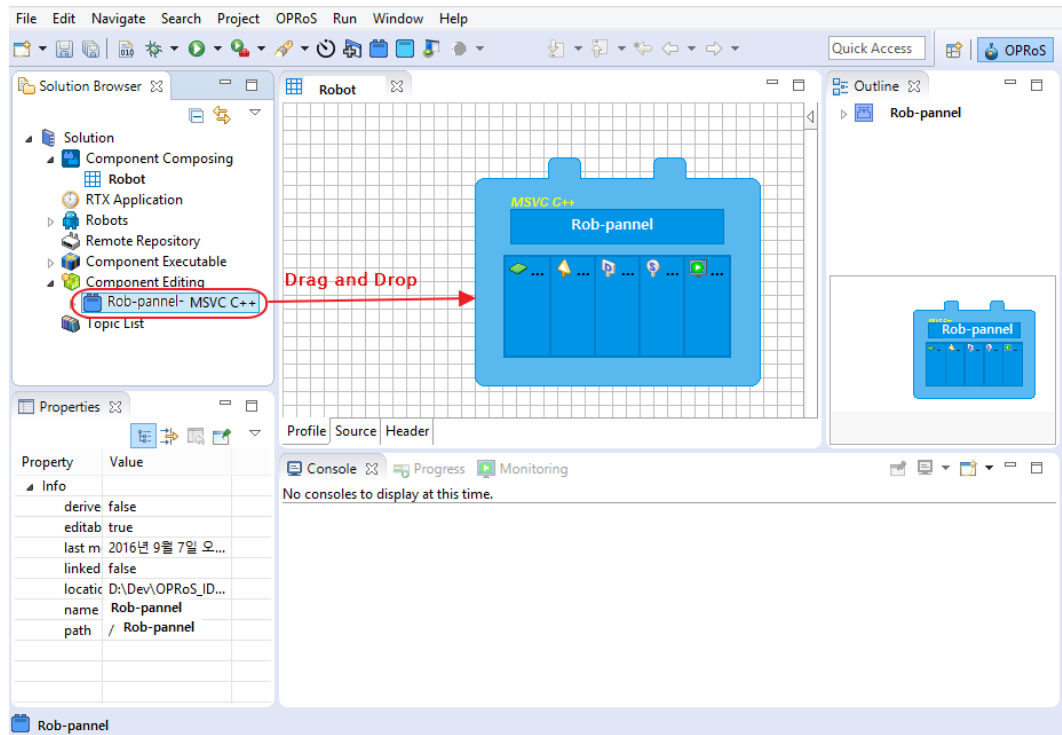
4. 메뉴에서 File > New > New Component를 선택하여 OPRoS 컴포넌트를 생성한다.

OPRoS 어플리케이션은 컴포넌트 profile(컴포넌트명.xml), 컴포넌트 바이너리(컴포넌트명.dll or so), 서비스 profile(제공된다면 서비스명.xml)로 구성되어 있다. 각각의 컴포넌트는 기능을 구현하거나 다양한 기능을 묶은 모듈 개념으로 이해하면 된다.



[그림 3-9] 새로운 컴포넌트 설정 화면

5. 생성된 OPRoS 컴포넌트(Rot-pannel)를 모눈으로 Drag&Drop하여 어플리케이션을 생성한다.



[그림 3-10] 컴포넌트 생성 화면

6. 생성된 OPRoS 컴포넌트(Rot-pannel)의 composing 화면에서 Header 및 Source를 클릭하여 파일을 수정한다.

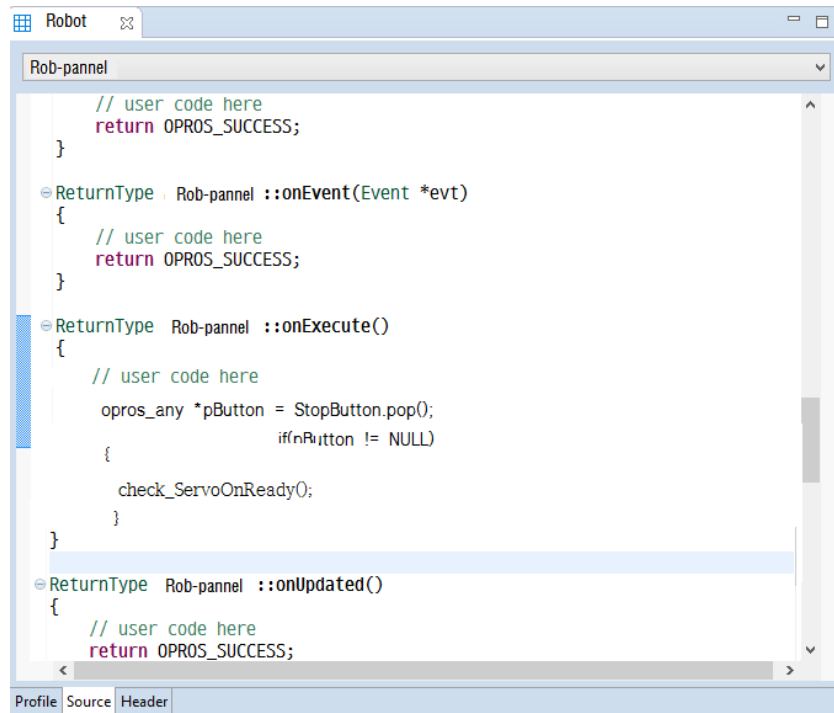
(1) 로봇의 긴급 정지 및 해지 기능 중 로봇의 패널(pannel)에서 수행되는 기능에 대해서는 컴포넌트에 Header 및 Source에 코드를 수정한다. 로봇의 긴급 정지 및 해지 기능 중 로봇의 배전반에서 수행되는 기능에 대해서는 별도의 컴포넌트를 만들어 생성된 컴포넌트의 Header 및 Source에 코드를 수정한다.

(2) 비상 정지 버튼이 회전하면 회전이 된 비상정지 버튼이 패널에 있는 것인지 조작반에 있는 것이 확인을 위한 코드를 작성한다.

(가) OPRoS 컴포넌트가 생성될 때 `onInitialize( )` → `onStart( )` 순으로 호출되고 종료될 때 `onStop( )` → `onDestroy( )` 순으로 호출된다. 따라서 이에 맞게 초기화 및 자원 해제를 수행한다.

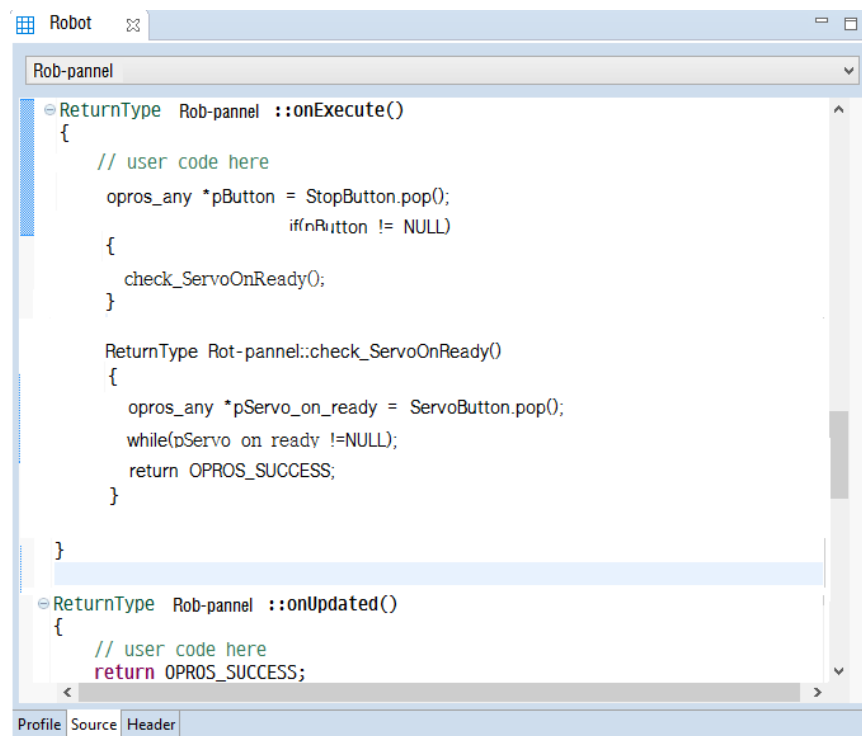
(나) `onExecute( )` 는 컴포넌트 profile의 execution type이 `periodic(default)`인 경우에 로봇의 배전반에서 수행되는 timer에 의해 주기적으로 호출되어 입력 상태를 확인한다.

(다) 변수를 선언하고 `onExecute( )` 내부에 실제 수행 코드를 작성한다.



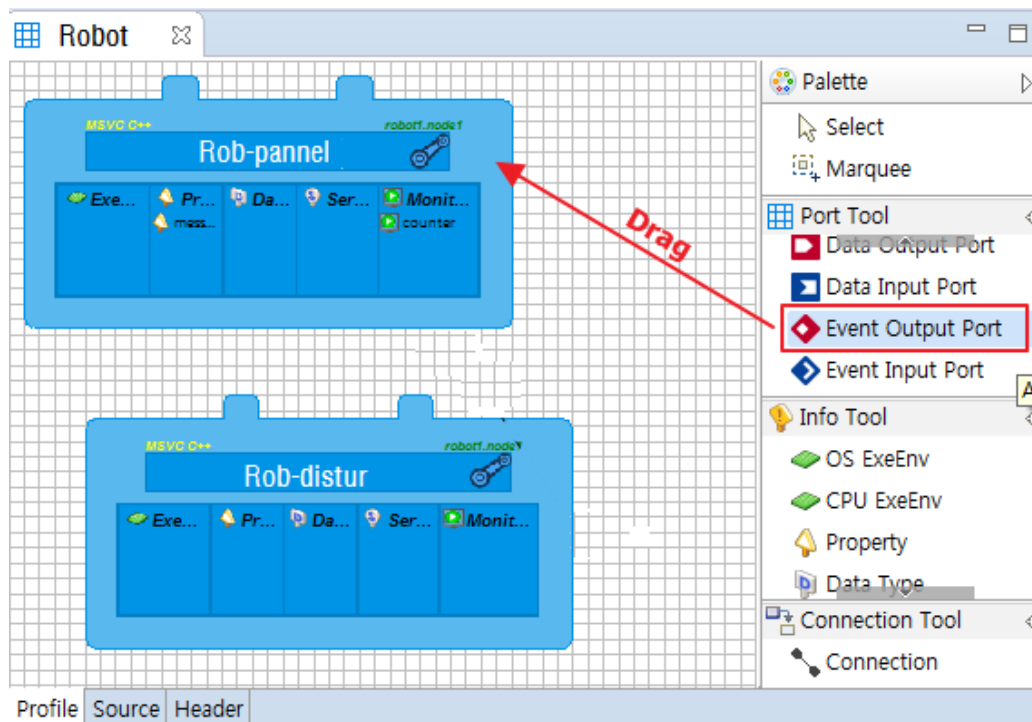
[그림 3-11] 로봇 패널의 코드 작성 화면

- (3) 패널의 버튼이 눌러졌다면 패널의 비상정지 버튼을 눌러졌다고 판단하고 서보 온 레디 버튼이 눌러졌는지 확인을 위한 check\_ServoOnReady( ) 함수 코드를 작성한다.

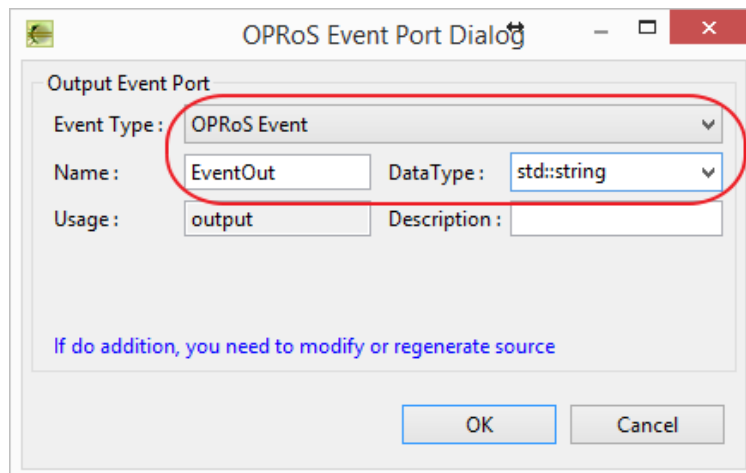


[그림 3-12] 로봇 패널 함수 코드 작성 화면

- (4) 서보 온 레디 버튼이 눌러졌다면 다시 정상적으로 동작을 하면 되고 눌러지지 않았다면 서보 온 레디 버튼이 눌러질 때까지 반복적으로 확인을 위한 코드를 작성한다.
7. 로봇의 긴급 정지 및 해지 기능은 주기적인 기능이 아니라 이벤트 성격의 기능으로 OPRoS의 이벤트 포트를 활용하여 미들웨어를 설계한다.
- (1) 로봇의 패널(Rob-pannel)에서 이벤트 포트를 추가하기 위해 Palette에서 Event Output Port를 끌어다 놓고 [그림 3-14]와 같이 설정한다.



[그림 3-13] 로봇 패널에 이벤트 설정 화면

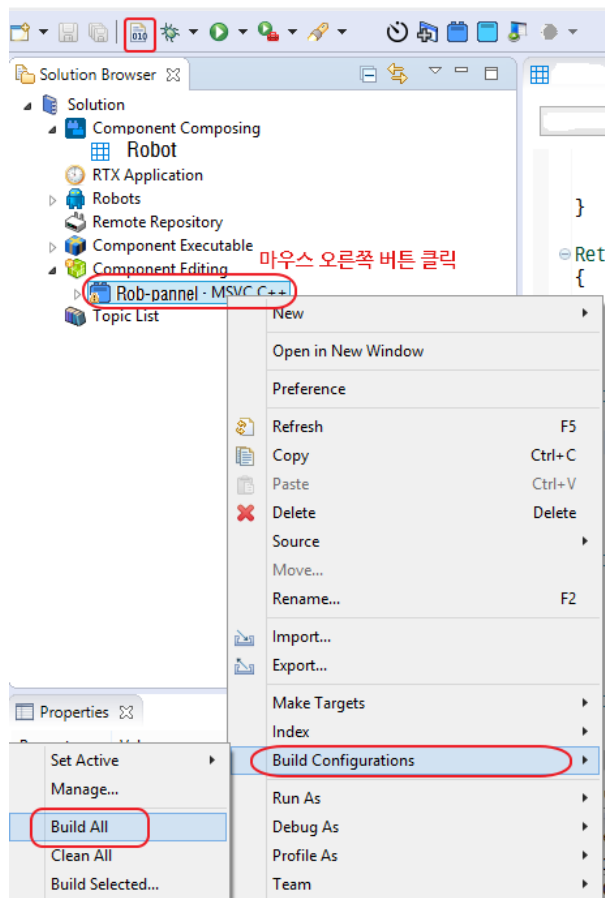


[그림 3-14] 로봇 패널에 이벤트 함수 설정 화면

(2) 로봇의 패널(Rob-pannel)에서 이벤트 정보를 주고받는 로봇 배전반(Rob-distur)의 경우도 동일하게 이벤트 포트를 추가하여 설정한다.

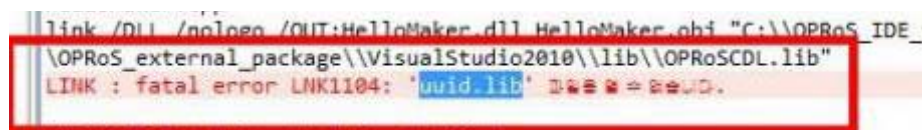
③ 작성한 코드를 빌드업하여 구현한다.

1. OPRoS의 Build Configuration에서 Release와 Debug 중 사용할 엔진을 Active로 설정한다.
2. OPRoS에서 빌드를 하기 위해서 Build All 버튼을 누르거나 Build Configurations에서 Build All을 클릭한다.



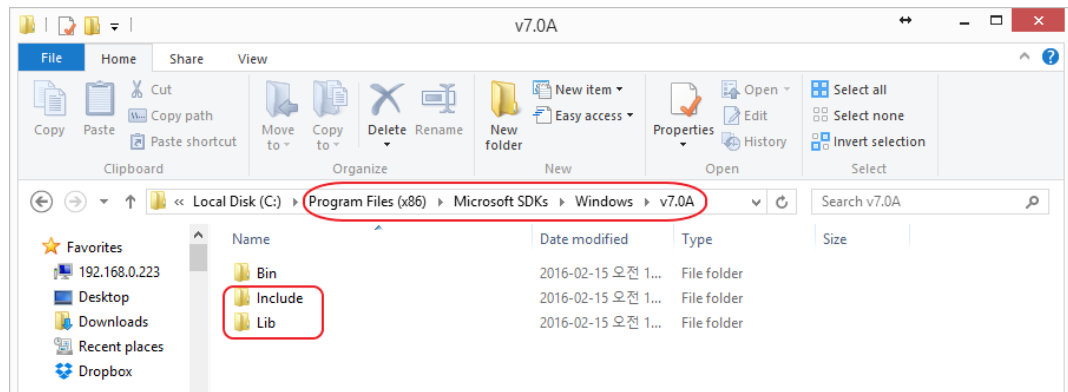
[그림 3-15] 로봇 패널에 Build All 동작 화면

3. 빌드 이후에 “uuid.lib” 가 없다고 메시지가 나타나면 개발도구 환경의 일부를 다음과 같이 수정한다.



[그림 3-16] uuid.lib 오류 화면

4. Solution Browser의 컴포넌트 위에서 마우스 오른쪽을 클릭해서 Properties를 선택하고 C/C++ Build > Environment 화면에서 include와 lib 경로 edit 버튼을 눌러 Microsoft SDKs > Windows > v7.0A 경로를 설정한다.



[그림 3-17] 윈도우 SDK 버전 경로 설정 방법 화면

### 수행 tip

- C 언어 관련 중급 이상의 지식을 가지고 있는 학생은 수업 내용을 이해할 수 있지만, C 언어에 대한 지식을 가지고 있지 않은 학생은 정상적으로 수업이 이루어 질수 없다.
- OPRoS에서 제공하는 다양한 자료를 활용하여 하나의 주제에 대해서 실습하는 것이 효율적이다.