차 례

로봇 작업 요구 사항 분석 및 소프트웨어 아키텍처 설계 학습모듈의 개요

학습모듈의 목표

로봇 기구 및 하드웨어의 특성을 이해하고 로봇의 동작 환경을 파악 및 분석한 이후에, 실제 로봇의 사용자 그룹별 로봇의 패턴 및 동작에 대한 요구 사항 분석할 수 있으며, 로봇 소프트웨어 아키텍처를 설계하기 위해서 요구 사항을 분석하고, 소프트웨어 아키텍처를 정의하고, 설계하고, 검증할 수 있다.

선수학습

로봇 공학, 소프트웨어 공학

학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 로봇 소프트웨어 요구 사항 정의하기	1-1. 로봇 소프트웨어 요구 사항 정의	1903080301_14v1.1	로봇 하드웨어 특성 분석하기
		1903080301_14v1.2	로봇 환경 분석하기
2. 로봇 소프트웨어 사용 사례 작성하기	2-1. 로봇 소프트웨어 사용 사례 작성	1903080301_14v1.3	사용자 분석하기
		1903080301_14v1.4	로봇 작업 시나리오 정의하기
3. 소프트웨어 아키텍처 정의하기	3-1. 로봇 소프트웨어 아키텍처 정의	1903080307_14v1.1	소프트웨어 아키텍처 요구 사항 분석하기
		1903080307_14v1.2	소프트웨어 아키텍처 정의하기
4. 소프트웨어 아키텍처 설계하기	4-1. 로봇 소프트웨어 아키텍처 설계	1903080307_14v1.3	소프트웨어 아키텍처 설계하기
		1903080307_14v1.4	소프트웨어 설계 검증하기

핵심 용어

컴포넌트 기반 개발, 소프트웨어 요구 사항 분석, 소프트웨어 요구 추출, 도메인 분석, 사용 사례 다이어그램, UML 모델링, 소프트웨어 아키텍처, 컴포넌트 다이어그램, 클래스 다이어그램, 상태 다이어그램

학습 1	**로봇 소프트웨어 요구 사항 정의하기**
학습 2	로봇 소프트웨어 사용 사례 작성하기
학습 3	로봇 소프트웨어 아키텍처 정의하기
학습 4	로봇 소프트웨어 아키텍처 설계하기

1-1. 로봇 소프트웨어 요구 사항 정의

학습 목표

- 로봇 하드웨어 구성도로부터 로봇의 제작 목표를 분석할 수 있다.
- 로봇 하드웨어의 주 기능과 제원을 분석할 수 있다.
- 각 하드웨어 자원의 스펙과 특징에 의한 소프트웨어의 요구 사항을 도출할 수 있다.
- 도출된 소프트웨어의 주요 고려 사항을 가지고, 소프트웨어 개발 항목(신규 또는 변경 필요)을 나열할 수 있다.
- 로봇의 동작 환경에 대해서 극한적인 환경, 일반적인 환경, 기타 환경으로 분류할 수 있다.
- 환경적인 요소가 로봇에 미치는 영향을 분석할 수 있다.
- 각 환경적인 요소가 소프트웨어의 동작에 미치는 영향에 대한 고려 사항을 도출할 수 있다.

필요 지식 /

1 컴포넌트 기반 개발(CBD, component based development) 방법론

1. 컴포넌트

로봇 소프트웨어의 경우 센서나 액추에이터 제어, 경로 계획, 장애물 감지 및 회피 등과 같이 매우 복잡한 기능들로 이루어져 있기 때문에 하나의 소프트웨어로 개발하는 것은 매우 어렵고 시간이 오래 걸리는 일이다. 이러한 이유로 소프트웨어를 각 기능별로 나누어서 개발하는 컴포넌트 기반 개발 방법론이 주로 이용된다.

컴포넌트란 재사용 가능한 독립적인 소프트웨어 구성의 단위를 의미한다. 컴포넌트에 대한 정의는 다양하지만 간단하게 정리한다면 독립적으로 개발 가능하며 배포 가능한 어플리케이션 블록으로 정의할 수 있다.

컴포넌트에 있어 가장 중요한 개념은 인터페이스이다. 인터페이스에 의한 컴포넌트 개발자와 컴포넌트 사용자 간의 상호 작용이 결정되며, 인터페이스를 통해서만 접근이 허용될 뿐 컴포넌트 내부는 엄격하게 은닉될 수 있기 때문이다.

로봇 소프트웨어의 경우 컴포넌트 기반 개발 방법론에 따라 각 기능들을 컴포넌트로 개발한 로봇 소프트웨어 아키텍처 또는 로봇 소프트웨어 프레임워크가 존재한다. 이러한 로봇 소프트웨어 아키텍처를 이용하는 경우 좀 더 쉬운 방법으로 로봇 시스템을 개발할 수 있다. 로봇 소프트웨어 아키텍처에 대해서는 3장을 참조한다.

2. 인터페이스

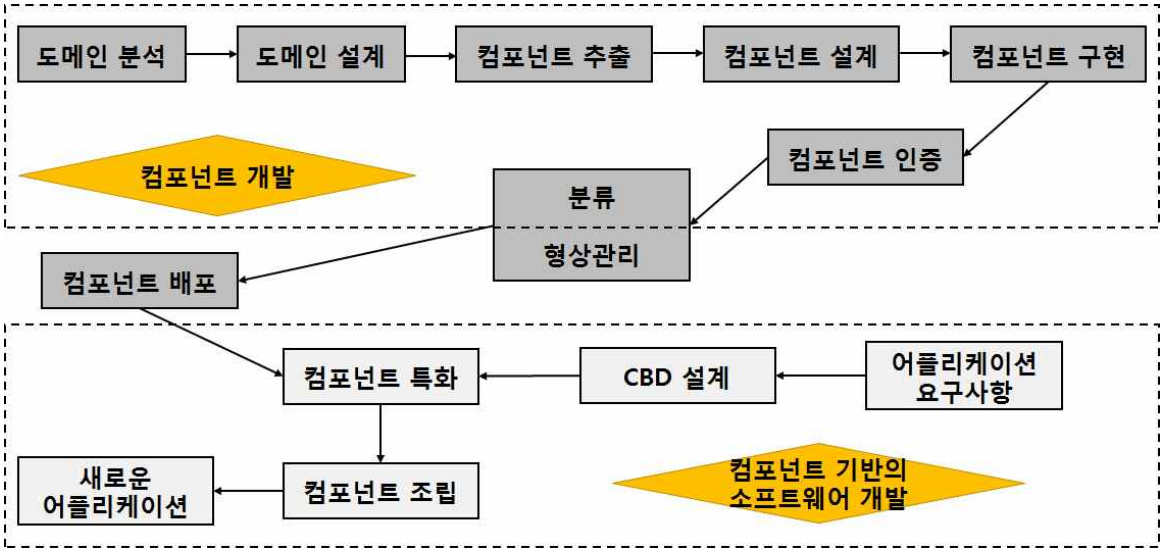
인터페이스란 컴포넌트가 수행할 하나 이상의 오퍼레이션의 집합이다. 인터페이스는 안정적이여야 하며 변화되지 않아야 한다. 하나의 컴포넌트는 하나 이상의 인터페이스를 가질 수 있으며 인터페이스별로 서로 다른 컴포넌트의 관점을 표현할 수 있다. 인터페이스는 여러 가지 장점을 갖는다. 인터페이스만 유지된다면 컴포넌트의 구현은 얼마든지 새로운 비즈니스 개념에 따라 바뀌어 질 수 있기 때문이다. 또 인터페이스만 결정이 되면 각각의 컴포넌트 개발은 완벽하게 병행 작업으로 이루어질 수 있게 된다. 컴포넌트는 인터페이스를 통해서만 상호 작용하므로 컴포넌트의 내부 구현을 알 필요도 알 수도 없기 때문이다. 또 인터페이스만 공유된다면 구현 언어를 자유롭게 선택할 수 있다.

3. 컴포넌트 기반 개발

컴포넌트 기반 개발 방법론은 [그림 1-1]과 같이 컴포넌트 개발과 컴포넌트 기반 소프트웨어 개발로 나눈다.

컴포넌트 개발은 전체 소프트웨어를 만드는 것이 아니라 개발하고자 하는 소프트웨어에 들어가는 서브프로그램을 개별로 만드는 것이다. [그림 1-1]의 상단 부분에 해당하는 것이 컴포넌트 개발에 필요한 일련의 업무들이다. 소프트웨어 컴포넌트는 각 비즈니스 컴포넌트가 운영될 수 있는 기반 아키텍처를 제공해야 하므로 해당 도메인에 대한 이해와 함께 기술 아키텍처에 대한 이해도 선행되어야 한다. 비즈니스 컴포넌트는 한 번만 사용될 목적으로 만드는 것이 아니라 다양한 소프트웨어 개발에 여러 번 재사용될 목적으로 만들어야 하기 때문에 해당 도메인에 대한 분석이 매우 중요하다.

컴포넌트 기반 소프트웨어 개발은 개발된 컴포넌트들을 조립하여 특정 소프트웨어를 개발하는 것을 말한다. 소프트웨어 개발에 있어서 가능한 부품 단위로 만들어서 개발하거나 혹은 이미 잘 개발되어진 부품들을 구매하여서 이러한 부품들을 조립하여 소프트웨어를 개발하자는 것이다. [그림 1-1]의 하단 부분이 컴포넌트 기반 소프트웨어 개발에 해당한다.



[그림 1-1] 컴포넌트 기반 소프트웨어 개발 방법론의 절차

2 소프트웨어 요구 사항 분석

요구 사항 분석은 소프트웨어 개발의 실제적인 첫 단계로 사용자의 요구에 대하여 이해하는 단계라 할 수 있다. 소프트웨어 개발은 기본적으로 광범위한 문제의 해결이라 할 수 있다. 문제를 해결하기 위해서는 먼저 사용자의 요구(requirement)를 이해하는 단계가 있어야 한다. 요구 사항의 성격과 범위를 이해해야 하며 문제 해결에 있어서의 제약 사항을 알아야 한다.

요구 사항 분석은 두 가지 단계로 구성된다. 현재의 상태를 파악하고 요구 사항을 정의한 후 문제 해결과 구현될 시스템의 목표를 명확히 도출한다. 이를 분석 단계라고 한다. 다음 단계는 명세서 작성 과정이다. 명세서에는 기술적 요구와 구현에 있어서 제약 조건 및 개발자와 사용자가 합의한 성능에 관한 사항도 명시되어야 한다. 명세서는 시스템 설계에서 참조될 사항을 포함하여 구현될 시스템에 대하여 전반적으로 알아야 할 사항을 포함하는 문서이다.

소프트웨어 요구 사항의 분석은 [그림 1-2]와 같이 4단계로 나눌 수 있다.

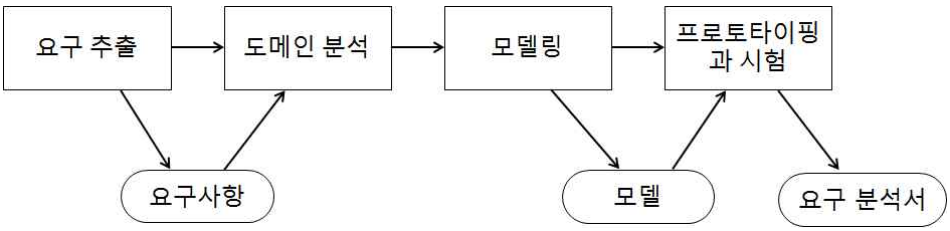
첫째, 요구 사항 추출은 소프트웨어 개발 계획 단계에서 정의한 문제의 범위 안에 있는 사용자의 요구를 찾는 일이다. 시스템의 중요한 목적을 만족시키기 위하여 시스템이 제공해야 할 사항을 찾는 것이라 할 수 있다. 특히 시스템이 제공하여야 할 기능적인 요구와 기능 이외의 조건들, 예를 들어 성능, 품질, 안전, 보안, 인터페이스 요구를 찾는다.

둘째, 도메인 분석은 문제 영역 안에 있는 중요한 사항들을 인식하기 위하여 정보를 수집하고 배경을 분석하는 단계이다. 인식되어야 할 중요한 기능이란 사용자의 요구와 관련된 객체(object), 관계(relation), 기능(function), 프로세스(process), 제약(constraints) 등이다. 도메인 분석은 사용자와의 인터뷰, 설문 조사, 실사 등을 통하여 현황을 파악하고 새로운 시스템

에 대한 요구를 모델링하기 위한 것이다.

셋째, 모델링은 도메인 분석을 통하여 찾아낸 중요한 개념, 특성, 관계에 대하여 개념화하는 작업이다. 주로 응용문제에 대한 정보를 수집하여 분석하고 이를 다이어그램을 이용하여 모델화하는 작업이라 할 수 있다.

넷째, 프로토타이핑과 시험은 분석된 기능적 요구의 타당성을 시험하기 위하여 프로토타입을 간단하게 만들어 본다. 이러한 결과를 걸쳐 소프트웨어 요구 분석서(software requirement specification)를 작성한다. 요구 분석서에는 시스템의 기능, 성능, 정보 표현, 제약, 검증 평가 기준에 대하여 기술하고 검토한다.



[그림 1-2] 요구 사항 분석 과정

3 소프트웨어 요구 추출

1. 소프트웨어 요구의 정의

소프트웨어 요구 사항(requirement)이란 시스템이 제공하여야 할 역량(capability)으로 정의할 수 있다. 소프트웨어 프로젝트는 시스템이 요구를 만족하고, 비용이 초과되지 않으며 일정 시간 내에 인도된다면 성공하였다고 본다. 따라서 진정한 요구를 찾는 일은 소프트웨어 프로젝트 성공을 위한 필수 조건이다.

요구 사항이란 시스템이 외형적으로 나타내는 기능이나 성능 등의 동작을 말한다. 예를 들어 모바일 로봇의 외형적 기능은 장애물 감지 기능이나 계단 등을 올라갈 수 있는 주행 기능 등이 될 수 있다. 모바일 로봇의 성능에 관한 기능의 경우 빠른 경로 탐색, 빠른 장애물 회피 등이 될 수 있다.

요구 사항의 종류에는 기능적 요구 사항과 비기능적 요구 사항으로 나눌 수 있다.

(1) 기능적(functional) 요구 사항

기능 요구 사항은 시스템과 외부 요소들 간의 인터랙션이라 할 수 있다. 기능적 요구를 결정하려면 시스템이 어떤 상태일 때 외부의 명령을 받아들여 어떤 반응을 하는지 기술하여야 한다. 예를 들어 모바일 로봇에서 장애물을 만나면 장애물을 감지하는 방법, 장애물을 회피하여 특정 목적지로 가기 위한 방법, 로봇의 위치를 알기 위한 방법 등이다.

(2) 비기능적(non-functional) 요구 사항

시스템이 수행하는 기능 이외의 사항, 즉 시스템 구축에 대한 성능, 보안, 품질, 안전 등에 대한 요구 사항들을 비기능적 요구라 한다. 예를 들어 1초 안에 경로를 탐색해야 한다거나, 로봇을 급정지시킬 수 있는 버튼이 있어야 한다거나, 원격으로 로봇의 경로를 보여줄 수 있어야 한다거나 등이다.

2. 요구 사항 추출

요구 추출(requirement elicitation)이란 사용자가 무엇을 원하는지 결정을 내리는 작업으로서 여러 가지 기법이 동원된다. 현재 판매되고 있는 로봇 시스템이 있다면 요구를 추출하는 작업은 비교적 수월하다. 그러나 아직 존재하지 않아 문제에 대한 해법을 생각해 보지도 않은 경우 사용자와 함께 아이디어와 해결책을 찾는 것이 어려울 수 있다.

요구를 추출하기 위하여 다음 세 가지 단계의 작업이 필요하다.

(1) 시스템에 대한 정보 출처 파악

요구 사항을 추출하기 위해서는 정보를 모으는 일부터 시작해야 한다. 다양한 형태의 정보원이 존재하며, 어떤 정보원으로부터 정보를 파악할 것인지를 결정해야 한다. 이러한 정보원에는 정부나 전문 기관의 법령이나 표준, 정부나 정부 산하 기관에서 발표되는 각종 정책이나 통계, 특허청에 등록된 특허, 전문 기관 등에서 출판되는 동향 분석 보고서나 시장 분석 보고서, 학술 기관 등에서 출판되는 논문과 세미나 자료, 전문가들의 의견이 들어 있는 인터넷 정보, 책과 같은 각종 출판물, 신문이나 방송의 각종 보도 자료 등이 있다.

(2) 요구 사항에 대한 정보 수집

요구 사항에 대한 정보를 모으는 방법에는 여러 가지가 있다. 고객이나 사용자의 발표를 직접 듣거나 인터뷰를 하는 방법, 문헌이나 인터넷 등을 직접 조사하는 방법, 설문지를 이용해 관련자나 전문가들의 의견을 취합하는 방법, 엔지니어들 간의 브레인스토밍 회의를 통해 요구 사항을 이끌어 내는 방법, 사용 사례를 찾아서 작성해 보는 방법 등이 있다.

(3) 요구 사항과 제한 사항의 정의

시스템의 요구란 주로 시스템이 수행하여야 하는 기능적 요구 사항과 비기능적 요구 사항, 성능적 요구 사항을 찾아내는 것이다. 이러한 요구들은 개발될 시스템의 기능이나 성능을 구체적으로 나타내게 된다.

수행 내용 / 로봇 소프트웨어 요구 사항 정의하기

재료 · 자료

- 로봇 하드웨어 구성도/로봇 작업 플로우차트
- 사용자 분석 보고서
- 로봇 동작 환경 매뉴얼
- 로봇에 대한 사용자 요구 사양서

기기(장비 · 공구)

- 컴퓨터, 프린터
- 사용자 요구 사항 분석용 소프트웨어
- 문서 작성용 소프트웨어

안전 · 유의 사항

- 로봇 소프트웨어를 개발함에 있어 로봇 하드웨어 구성도 및 로봇 기구 구성도 등 로봇 시스템의 요구 사항을 면밀히 분석하여 로봇 소프트웨어의 요구 사항을 도출해야 한다.
- 로봇 소프트웨어의 요구 사항을 도출할 때 참고할 수 있는 상용 로봇 시스템에 대한 자료를 최대한 많이 확보해둘 필요가 있다.

수행 순서

[1] 로봇 시스템 개발 기획 보고서를 검토한다.

1. 개발 기획 보고서에서 정의되어 있는 로봇 시스템의 개발 목표를 검토한다.
   로봇 시스템의 개발 목표를 검토하고, 개발하고자 하는 목표에 대한 세부 목표를 세운다. 세부 목표 설정은 개발자의 경험에 따라 좌우되는 경우가 많으므로, 여러 분야의 개발자 간의 브레인스토밍을 통하여 합리적이라고 할 수 있는 세부 목표를 설정한다.
2. 개발 기획 보고서에 서술되어 있는 시장 및 기술을 검토한다.
   개발 기획 보고서에 서술되어 있는 시장의 흐름과 기술 동향을 검토하고 개발 기술에 대한 방향성을 확보하도록 한다. 최근 급격한 기술 발전 움직임에 대응한 개발 플랫폼을 검토하고 자사 및 경쟁사 그리고 시장의 분석을 진행한다. 특히 최근 로봇 하드웨어 및 로봇 소프트웨어의 발전에 유의하고 이러한 움직임에 맞추어 개발 방향을 잡을 수 있도록 한다.

3. 관련 부서와 협의하여 개발 일정을 확인한다.

개발하고자 하는 목표에 대한 세부 일정 등을 확인한다. 특히 개발 품질의 신뢰성 확보를 위한 테스트 기간 등의 확인을 통하여 자원 분배 및 기능 구현에 따른 개발 일정 확인을 하도록 한다.

2 개발 기획 보고서로부터 소프트웨어 요구 사항 후보들을 선별한다.

1. 개발 기획 보고서에서 로봇 하드웨어 구성도 등을 분석한다.

개발 기획 보고서 작성 과정에서 수집된 요구 사항들과 로봇 기구부 설계안, 로봇 하드웨어 구성도 등을 검토한 후 소프트웨어 요구 사항 후보들을 선별한다. <표 1-1>은 재난 구조용 모바일 로봇을 위한 요구 사항 정리표를 나타내고 있으며, [그림 1-3]은 재난 구조용 모바일 로봇의 예를 나타내고 있다. 이러한 자료들을 분석한 후에 로봇 소프트웨어에 대한 요구 사항을 추출해 내어야 한다.

<표 1-2> 재난 구조용 모바일 로봇을 위한 요구 사항 정리표의 예

번호	요구 사항	유형 분석			중요도	난이도	비고
		기능	비기능	성능			
1	장애물이 있는 작업 공간에서 100m를 움직이는 데 10분 이내의 시간이 걸려야 한다.	0			중	상	
2	10Kg의 작업물을 잡을 수 있는 로봇 팔이 있어야 한다.	0			중	하	
3	작업 공간의 경우 돌덩이나 깨진 건물 잔해 등이 있을 수 있다.			0	중	하	
4	30° 의 경사로를 올라갈 수 있어야 한다.	0			상	상	
5	로봇은 두 사람이 들어 움직일 수 있는 무게여야 한다.		0		하	중	
6	사람이나 장애물과 부딪쳤을 때에는 정지할 수 있어야 한다.			0	중	중	
7	사람과 장애물을 구분할 수 있어야 한다.		0		하	중	
8	....						

출처: SuperDroid HD2-S Mastiff Tactical Robot(http://www.robotshop.com). 2016. 9. 30. 스크린샷.
[그림 1-3] 재난 구조용 모바일 로봇의 외형 예

2. 로봇 소프트웨어 요구 사항을 추출한다.

로봇 기구부 설계안, 로봇 하드웨어 구성도 등으로부터 <표 1-2>와 같은 로봇 소프트웨어 요구 사항을 추출해낸다. 로봇 소프트웨어에 대한 요구 사항을 분석할 때에는 상용 로봇 시스템이나 각종 문헌 자료 등을 통하여 어떠한 기능들이 필요한지에 대해 미리 이해를 해 두어야 한다.

<표 1-2> 재난 구조용 모바일 로봇을 위한 소프트웨어 요구 사항

번호	로봇 시스템에 대한 요구 사항	소프트웨어 요구 사항	비고
1	장애물이 있는 작업 공간에서 100m를 움직이는 데 1.5분 이내의 시간이 걸려야 한다.	로봇 주행 로봇 경로 계획 로봇 자기 위치 인식 모터 제어	
2	10Kg의 작업물을 잡을 수 있는 로봇 팔이 있어야 한다.	매니퓰레이터 제어 그리퍼 제어	
3	작업 공간의 경우 돌덩이나 깨진 건물 잔해 등이 있을 수 있다.	캐터필더 슬립 억제 한쪽 바퀴가 들리거나 뒤집어지려고 할 때 자세 유지	
4	30° 의 경사로를 올라갈 수 있어야 한다.		
5	로봇은 2사람이 들어 움직일 수 있는 무게여야 한다.		
6	사람이나 장애물과 부딪쳤을 때에는 정지할 수 있어야 한다.	장애물 회피 및 긴급 정지	
7	사람과 장애물을 구분할 수 있어야 한다.	장애물 감지	
8	....		

3. 로봇 소프트웨어에 대한 기능을 결정한다.

로봇 소프트웨어에 대한 요구 사항을 추출한 후에 반드시 필요한 기능과 그렇지 않은 기능을 구분한 후, 개발하여야 할 로봇 소프트웨어에 대한 기능을 결정한다.

[그림 1-4]는 경량 팔을 가진 재난 구조용 모바일 로봇에서 구현되어야 할 기능들의 목록을 표시해 두었다. 이러한 기능들은 로봇 소프트웨어 컴포넌트로 개발될 것이다.

[그림 1-4] 재난 구조용 모바일 로봇의 소프트웨어 기능

수행 tip

- 로봇 소프트웨어 요구 사항은 로봇 소프트웨어 아키텍처를 개발할 때 컴포넌트로 고려되어야 한다. 따라서 가능한 한 모든 기능에 대해 검토해두어야 한다.

2-1. 로봇 소프트웨어 사용 사례 작성

학습 목표

- 로봇을 동작시키고 사용하는 사용자 그룹을 사용 목적별, 연령별, 사용 방법의 전문성별 등으로 분류할 수 있다.
- 사용자의 특성이 소프트웨어의 설계에 미치는 영향을 분석 할 수 있다.
- 사용자의 특성이 소프트웨어의 설계에 미치는 영향에 대한 고려 사항을 도출할 수 있다.
- 하드웨어 특성, 동작 환경, 사용자 환경 등에 따른 로봇 작업 시나리오를 정의 할 수 있다.
- 작성된 로봇 작업 시나리오에 각종 예외 사항, 특이 사항, 조건 이벤트 사항 등을 추가할 수 있다.
- 도출된 상세 로봇 시나리오별로 현 로봇 소프트웨어의 신규 또는 변경 개발 항목을 도출할 수 있다.

필요 지식 /

1 도메인 분석

요구 사항 추출 이후에는 요구들을 바탕으로 한 도메인 분석 단계로 진행한다. 도메인이란 해당 소프트웨어의 적용 범위 또는 동작 범위를 말한다. 그러므로 도메인 분석이란 로봇 소프트웨어가 적용되는 범위와 관련된 요구들의 배경과 환경을 분석하여 개발자와 고객 간의 공통 개념을 세워가는 과정에 해당된다.

도메인 분석에는 도메인에서 사용되는 개념들을 정리한 도메인 사전, 시스템에서 반드시 사용해야 하는 공식이나 시스템의 사용 규칙 등을 정리한 비즈니스 규칙이 포함된다.

1. 도메인 사전

고객과 개발자 간의 의사소통을 위하여 도메인 내에서 사용되는 용어들을 정의할 필요가 있다. 용어 사전의 내용은 쉽고 정확하여야 한다. 용어 사전을 구성하는 구성 요소는 명칭, 타입, 설명 또는 예시이다.

(1) 명칭

각각의 명칭들은 사용되는 개념들에 대한 식별자이다. 일반적인 사전에서와 같이 하나의 명칭은 여러 가지의 의미를 가질 수 있다. 국어사전의 단어라고 보면 된다.

(2) 타입

각각의 명칭들은 사용되는 개념들에 대한 식별자이다. 일반적인 사전에서와 같이 하나의 명칭은 여러 가지의 의미를 가질 수도 있고, 명사, 동사 등 쓰임새가 다를 수도 있다. 타입은 기능, 역할, 개념, 기기, 사람 등으로 분류한다.

(3) 설명 또는 예시

용어를 정확하게 설명하기 위하여 사용되는 예시나 추가 설명을 기입하면 좋다.

2. 비즈니스 규칙

비즈니스 규칙이란 해당 시스템이 동작하는 데 있어서 반드시 지켜져야 할 규칙, 정책을 말한다. 비즈니스 규칙은 고객, 매뉴얼, 업무 지시서, 전문가 인터뷰 등을 통하여 수집된다.

비즈니스 규칙이 일목요연하게 관리되어 있지 않으면, 개발자가 규칙을 나중에 발견하게 되었을 경우 많은 수정 작업을 거쳐야 할 수도 있다. 또 문서화되어 있지 않은 여러 가지 비즈니스 규칙을 잘 아는 기존의 개발자가 퇴사를 하여 개발자가 바뀌는 경우에도, 새로운 개발자는 로봇 소프트웨어에 녹아 들어있는 비즈니스 규칙을 이해하고 파악하기 위하여 많은 시간 투자를 해야 할 것이다.

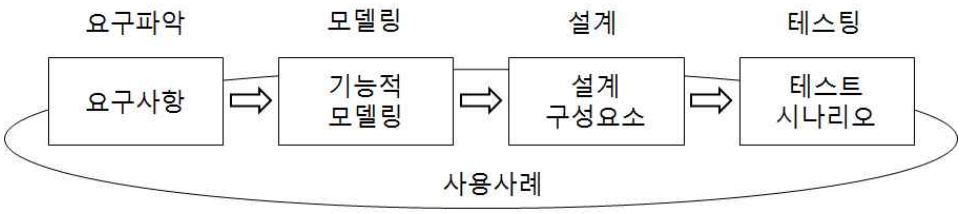
2 사용 사례

1. 사용 사례

성공적으로 개발된 로봇 소프트웨어는 사용자의 요구 사항을 모두 반영하여야 한다. 사용자의 요구 사항은 말, 그림, 매뉴얼, 기존의 유사 시스템, 개발 요구서, 사용 사례 등을 통하여 개발자에게 전달된다. 개발자는 사용자의 요구 사항을 분석하여 시스템이 제공해야 할 기능들을 구현하고, 구현된 기능들을 확인하는 일을 하게 된다.

개발 요구서가 정확하게 주어지지 않는 한 개발자는 사용자의 요구 사항을 분석하기 위하여 여러 방법을 사용한다. 사용 사례란 시스템이 수행할 것으로 기대되는 기능을 말한다. 사용 사례는 사용자 또는 외부 시스템이나 기타 요소들이 시스템과 상호 작용하는 흐름을 모델링한 것이다. 사용 사례가 중요한 이유는 [그림 2-1]과 같이 시스템 개발을 시작하는

요구 분석 단계에서부터 설계 단계를 거쳐 테스팅에 이르기까지 꼭 필요한 길잡이라고 할 수 있다. 즉 모델링 단계에서는 기능적 모델링을 할 수 있는 단초를 제공해 주며, 설계 단계에서는 상세한 설계 구성 요소가 되며, 테스팅 단계에서는 테스트 시나리오로 사용할 수 있기 때문이다.



[그림 2-1] 사용 사례의 적용 범위

사용 사례에서 시스템과 상호 작용하는 객체를 액터(actor)라 한다. 각 사용 사례는 외부 객체들이 시스템과 어떻게 상호 작용하는지 가능한 시나리오를 나타내는 것이다. 사용 사례는 관련된 액터와 함께 시스템이 어떻게 기능하는지를 텍스트 형태로 기술할 수도 있다. 각 사용 사례에 대하여 발생할 수 있는 모든 이벤트를 순서대로 파악하고 이에 대한 시스템의 반응들을 나열한다. 시스템 밖에 있는 요소들이 시스템을 사용하는 사용 사례를 모두 모으면 시스템의 기능을 설명하는 것이 될 것이다.

개발자는 사용 사례를 분석하여 사용자의 요구 사항을 정리하고, 이를 바탕으로 구현되어야 할 기술 내용을 결정할 수 있다. 시스템의 구현이 잘 되었는지를 평가하기 위하여 각각의 사용 사례를 이용하여 시스템의 동작을 확인한다.

사용 사례는 충분히 많이 작성하여야 하며, 개발자가 로봇 소프트웨어를 만들 때에 고려된 적이 없는 상황이 생기지 않도록 하여야 한다. 그러므로 사용자가 해당 시스템을 사용할 때에 생길 수 있는 모든 상황들을 기술할 수 있도록 노력하여야 한다.

2. 사용 사례의 용어

(1) 액터

액터(actor)란 시스템과 작용하는 외부의 객체로서, 사람 또는 다른 시스템이다. 재난 구조용 모바일 로봇의 예를 들어보자. 액터가 사람인 예는 모바일 로봇을 감시하거나 제어하는 사용자가 될 수 있다. 액터가 사물인 예는 움직이거나 정지해 있는 장애물이 될 수 있다. 액터가 시스템인 경우는 두 대의 모바일 로봇이 상호 협력 작업을 해야 되는 경우에서 상대 모바일 로봇이 될 수 있다.

(2) 시나리오

시나리오는 사용자의 관점에서 액터와 시스템 사이에 일어나는 일들을 기술한 것이다. 여러 가지 시나리오들을 분석하여 일반화된 사용 사례를 찾는다. 시나리오는 정상적인 흐름 외에도 오류, 특정한 장소와 시간 등의 예외적인 흐름도 포함한다.

(3) 사용 사례

사용 사례는 하나의 시나리오 또는 여러 개의 시나리오를 종합하여 일반화한 것이다. 사용 사례는 개발자와 고객 간의 의사소통 수단이다. 예외적인 사건의 흐름에 대하여 개발자가 알 수 있게 하고, 개발된 시스템의 기능에 대하여 고객이 알 수 있게 된다.

3. 사용 사례의 표현

사용 사례 다이어그램은 시스템의 기능을 쉽게 알아볼 수 있도록 사용 사례를 그림으로 나타낸 것이다. 시스템이 커지면 커질수록 각각의 사용 사례들은 중복되거나 복잡해지는데, 사용 사례 다이어그램은 이런 문제를 해결하는 데 도움이 된다. 사용 사례 다이어그램은 액터, 사용 사례, 관계의 세 가지로 구성된다.

사용 사례 다이어그램은 사용 사례를 액터의 관점에서 기술하고, 시스템 내부의 동작이 아닌 액터와 사용 사례의 상호 작용을 나타내는 것이 좋다.

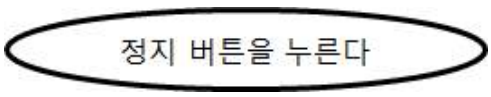
(1) 액터

작업을 수행하기 위하여 시스템의 지원을 받는 객체는 액터가 될 수 있다. 시스템의 주요 기능을 사용하는 객체는 액터가 될 수 있다. 유지보수와 관리 같은 부수적인 기능을 사용하는 객체는 액터가 될 수 있다. 다른 외부 하드웨어나 소프트웨어와 연결되어 동작한다면 그 시스템은 액터가 될 수 있다. [그림 2-2]는 액터의 예를 나타낸다.

[그림 2-2] 액터의 예

(2) 사용 사례

사용 사례 다이어그램의 사용 사례는 시스템이 제공하는 기능이나 서비스이다. 사용 사례는 액터의 입장에서 보는 시스템의 동작을 기술한 것이다. [그림 2-3]은 사용 사례의 예를 나타낸다. 사용자가 원격 모니터링 및 제어 장치를 이용하여 위험 상황을 감지하고 정지 버튼을 누르는 기능을 사용 사례의 예라 할 수 있다.



[그림 2-3] 사용 사례의 예

(3) 관계

(가) 연결 관계

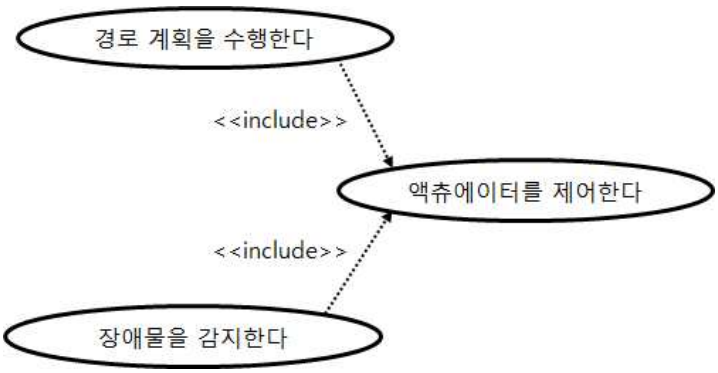
액터와 상호 작용하는 사용 사례는 실선 화살표로 연결된다. 하나의 액터는 복수의 사용 사례와 연결할 수 있다. [그림 2-4]는 액터와 사용 사례의 연결 관계를 나타내고 있다.

[그림 2-4] 연결 관계의 예

(나) 포함관계

여러 개의 사용 사례에서 공통적으로 사용되는 사용 사례를 떼어 냄으로써 사용 사례 다이어그램의 복잡도를 줄일 수 있다. 이때 여러 개의 사용 사례와 떼어 낸 사용 사례는 서로 포함관계에 있다고 말한다. [그림 2-5]에서 보듯이 사용 사례 다이어그램에서는 점선 화살표로 표시하고 <<include>>라는 표시를 한다.

포함된 사용 사례는 포함하는 사용 사례가 수행될 때 반드시 같이 수행되어야 한다. 그림의 사용 사례의 의미는 사용자가 정지 버튼을 누르거나, 로봇이 초음파 센서를 이용하여 장애물을 감지하였을 때는 액추에이터를 제어한다는 의미이다. 액추에이터를 제어한다는 사용 사례에는 여러 가지 경우의 수에 따라 정지하는 기능, 왼쪽 또는 오른쪽으로 방향을 전환하는 기능, 후진을 하는 기능 등이 포함될 수 있다.



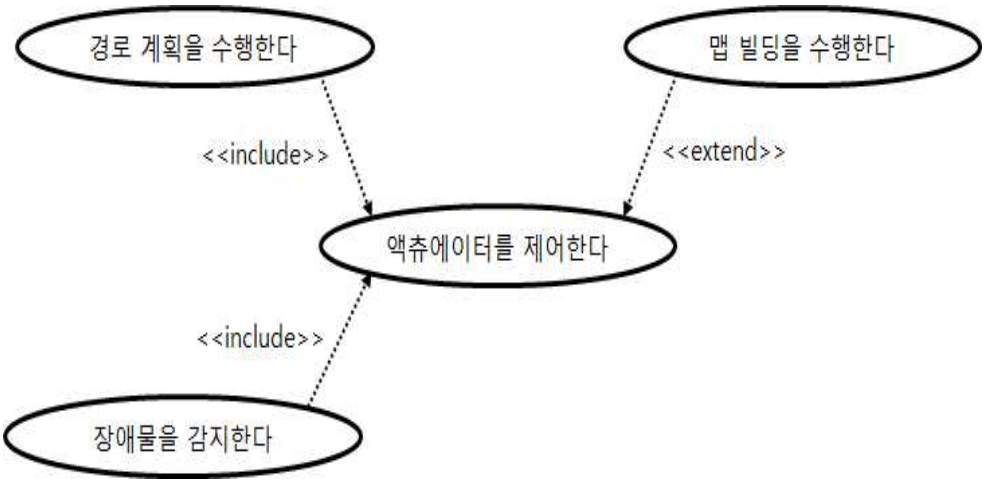
[그림 2-5] 포함 관계의 예

(다) 확장 관계

하나의 사용 사례는 특별한 조건 하에 다른 사용 사례로 확장될 수 있다. 포함 관계에 있는 2개의 사용 사례는 반드시 같이 수행되는 반면, 확장된 사용 사례는 예

외적으로 이벤트가 추가될 때에만 수행된다.

[그림 2-6]에서 보듯이 사용 사례 다이어그램에서는 방향이 반대인 점선 화살표로 표시하고 <<extend>>라는 표시를 한다. 이때 조건을 같이 명시하면 읽기 좋은 사용 사례 다이어그램이 된다. 장애물이 감지되어 액추에이터 제어를 할 경우에는 경로 계획을 수행하여 최적의 경로를 생성하게 된다. 만약, 감지된 장애물이 맵을 재생성해야 할 만큼 중요한 장애물이라면 맵 빌딩을 수행하여야 한다. 아래 그림에서 맵 빌딩은 조건이 성립되었을 때 수행하면 된다는 의미이다.



[그림 2-6] 확장 관계의 예

3 모델링

1. 객체 지향 프로그래밍(object-oriented programming)

객체 지향 개념에서는 소프트웨어를 여러 개의 객체 모임으로 생각하며 프로그래밍을 하는 방법이다. 여기에서 객체는 데이터와 관련 함수를 모아 놓은 것이다. 즉, 관련된 자료와 함수를 객체로 묶어 놓고 이들의 상호 작용에 의하여 작업이 수행된다. 객체 지향 개념은 다음 세 가지를 기본 요소로 한다.

(1) 객체(object)

객체는 우리 주변에 존재하거나 생각할 수 있는 것을 말한다. 예로 모바일 로봇에서 사람으로는 사용자 등이 있으며, 대상으로는 장애물, 작업물 등이 있으며, 장치로는 액추에이터, 센서, 로봇 팔 등이 있다.

(2) 클래스(class)

클래스는 비슷한 특성을 가진 객체들을 그룹화시킨 틀이며, 유사한 객체들의 집단에 대해 표현, 모델화, 추상화한 개념이다. 즉, 유사한 속성, 공통된 행위, 공통된 관계성, 공통된 의미를 가지는 객체들의 집단에 대한 표현 방법이다. 예로 모바일 로봇에서 사

용자의 행동들, 센서의 결과값에 따른 액추에이터 등을 속성(데이터)과 행위(코드)로 추상화시켜 클래스로 만든다.

클래스는 객체 지향 소프트웨어 작성의 가장 기본적인 단위로서, 속성(또는 프로퍼티, attribute, property)과 메서드(또는 행위, method, behavior)로 구성된다.

(3) 메시지(message)

객체들은 각각 독립적으로 존재하지만, 다른 객체와 상호 작용하면서 소프트웨어를 운영한다. 이때 객체들 간에 상호 작용을 위한 수단이 메시지이다. 메시지는 어떤 한 객체가 다른 객체에게 특정 작업을 요청하는 신호이다.

2. 통합 모델링 언어(UML, unified modeling language)

통합 모델링 언어, 즉 UML은 객체 관리 그룹(object management group)에서 관리하고 있는 소프트웨어 프로그램을 위해 사용되는 표준화된 범용 모델링 언어이다.

UML은 BOOCH의 booch method, RUMBAUGH의 OMT(object modeling technique), JACOBSON의 OOSE(object-oriented software engineering)에 의해 소개된 모델 표기법을 단일화시킨 모델링 언어이다.

객체 관리 그룹은 1989년에 설립된 비영리 단체로서, 1997년 1월 IBM, HP, MICROSOFT, ORACLE 등 여러 업체들이 참여하여 UML 버전 1.0을 제출하였으며, 2003년 6월 실시간 시스템과 워크플로우 시스템을 모델링할 수 있는 버전 2.0이 발표되었다.

UML은 객체 지향 설계를 위한 표준 언어로서, 소프트웨어 시스템의 산출물을 가시화, 명세화, 구축, 문서화하는 데 사용된다.

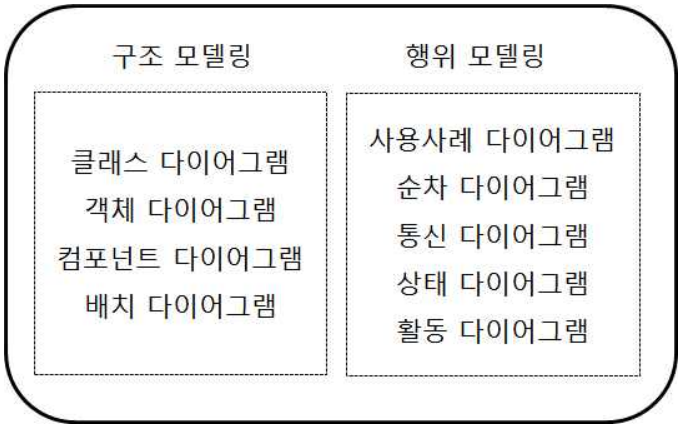
- UML은 소프트웨어 시스템을 시각적으로 표현하는 시각화 언어이다.
- UML은 소프트웨어 시스템의 구조를 명세화하는 명세화 언어이다.
- UML은 소프트웨어 시스템을 구축하는 데 사용되는 구축 언어이다.
- UML은 소프트웨어 시스템을 문서화하는 문서화 언어이다.

3. UML 다이어그램의 종류

UML 다이어그램에는 [그림 2-7]과 같이 시스템의 정적인 부분(구조 모델링)을 가시화하기 위하여 클래스 다이어그램, 객체 다이어그램, 컴포넌트 다이어그램, 배치 다이어그램 등을 사용하며, 시스템의 동적인 부분(행위 모델링)을 가시화하기 위하여 사용 사례 다이어그램, 순차 다이어그램, 통신 다이어그램, 상태 다이어그램, 활동 다이어그램 등을 사용한다.

(1) 클래스 다이어그램(class diagram)

클래스 다이어그램은 객체 지향 프로그램의 가장 근간이 되는 다이어그램으로 시스템의 정적인 구조를 나타낸다. 클래스 다이어그램은 시스템 내부에서 속성과 메서드들을 가진 클래스들과 그들 간의 관계를 표시한다. 두 클래스 간에 관계가 있다는 것은 이들이 하나 이상의 사용 사례(use case)를 수행하는 데 있어서 상호 협동하는 것을 의미한다.



[그림 2-7] UML 다이어그램의 종류

(2) 객체 다이어그램(object diagram)

객체 다이어그램은 객체와 객체들 사이의 관계를 나타낸다. 객체 다이어그램은 특성 시점의 객체들의 구조적 상태를 표현한다.

(3) 컴포넌트 다이어그램(component diagram)

컴포넌트 다이어그램은 컴포넌트 사이의 구성과 의존을 나타낸다. 컴포넌트 다이어그램은 클래스 다이어그램과 관련이 있는데, 일반적으로 하나의 컴포넌트는 클래스 다이어그램에 있는 하나 또는 그 이상의 클래스, 인터페이스, 통신들과 대응된다.

(4) 배치 다이어그램(deployment diagram)

배치 다이어그램은 시스템이 실행될 때 처리하는 노드와 그 노드에 있는 컴포넌트들의 구성을 나타낸다. 배치 다이어그램은 컴포넌트 다이어그램과 관련이 있는데, 일반적으로 하나의 노드는 컴포넌트 다이어그램 안에 있는 하나 또는 그 이상의 컴포넌트를 수용하기 때문이다.

(5) 사용 사례 다이어그램(use case diagram)

사용 사례 다이어그램은 액터와 사용 사례를 이용하여 시스템의 기능을 나타내기 위하여 사용자의 요구를 추출하고 분석하는 데 사용된다. 사용 사례 다이어그램은 사용 사례와 액터(actor), 이들 간의 관계를 표현한다.

(6) 순차 다이어그램(sequence diagram)

순차 다이어그램은 클래스 상의 메시지 교환을 시간의 흐름에 따라 나타낸 다이어그램이다. 순차 다이어그램은 객체들이 참여한 시간적, 순서적 처리 흐름을 동적으로 표현한다. 시간이 지남에 따라 객체 간에 어떤 상호 작용이 발생하였는지를 객체 간에 교환되는 메시지를 중심으로 표현한다.

(7) 상태 다이어그램(state diagram)

상태 다이어그램은 외부 자극에 대한 시스템의 동적 상태 변화를 나타낸다. 상태 다이

어그램은 외부 이벤트에 대하여 민감하게 상태를 변화시키는 객체를 모델링한다. 상태(state), 전이(transition), 사건(event)으로 구성되어 있으며, 사건에 따라 순차적으로 발생하는 객체의 행동에 중점을 둔다.

(8) 활동 다이어그램(activity diagram)

활동 다이어그램은 시스템 내부에 있는 활동의 흐름을 나타낸다. 활동 다이어그램은 유즈 케이스의 이벤트 흐름을 표현하는 데 적당하다.

수행 내용 / 로봇 소프트웨어 사용 사례 작성하기

재료 · 자료

- 로봇 하드웨어 구성도/로봇 작업 플로우차트
- 사용자 분석 보고서
- 로봇 동작 환경 매뉴얼
- 로봇에 대한 사용자 요구 사양서

기기(장비 · 공구)

- 컴퓨터, 프린터
- 사용자 요구 사항 분석용 소프트웨어
- 문서 작성용 소프트웨어

안전 · 유의 사항

- 고객과 개발자가 사용하는 언어나 표현 방식, 이해 내용이 서로 다를 수 있으나, 서로가 이해할 수 있도록 도메인 사전을 쉽고 정확하게 작성한다.
- 비즈니스 규칙이 일목요연하게 관리되어 있지 않으면, 개발자가 규칙을 나중에 발견하게 되었을 경우, 최초의 개발보다 더 많은 비용과 노력을 필요로 하게 됨을 유의한다.
- 사용 사례를 작성할 때에는 개발자와 고객 상호간의 입장을 존중하여야 한다. 또 사용 사례는 한 번에 완성되는 것이 아니라 고객과 개발자가 다듬어 완성하여야 한다.
- 사용 사례는 잘못 작성되었을 경우, 전체 개발 작업에 가장 큰 역효과를 미치게 된다.

수행 순서

1 로봇 소프트웨어를 위한 도메인 분석을 수행한다.

1. 로봇 소프트웨어를 위한 도메인을 정의한다.

도메인이란 해당 소프트웨어의 적용 범위 또는 동작 범위를 말한다. 가장 먼저 개발자와 고객 간에 적용 범위나 동작 범위에 대한 용어들을 정의하여야 한다.

<표 2-1>과 같이 로봇 소프트웨어를 위한 도메인을 정의하고, 각 도메인에서 수행해야 할 기능의 범위를 정한다.

<표 2-1> 재난 구조용 모바일 로봇을 위한 도메인과 범위의 예

도메인	범위
주행	액추에이터 제어(actuator control) 스티어링(steering) 자기 위치 확인(localization) 경로 계획(path planning) 맵 빌딩(map building)
장애물	초음파 센서 인터페이스(ultrasonic sensor interface) 장애물 감지(obstacle detection) 장애물 회피(obstacle avoidance)
작업	작업 할당(task definition) 매니퓰레이터 제어(manipulator control) 이미지 프로세싱(image processing)
사용자	원격 모니터링-비디오 및 오디오 디스플레이(video or image display) 원격 모니터링-로봇 상태 디스플레이(robot health display) 원격 모니터링-센서 데이터 디스플레이(sensor data display) 원격 제어(tele-operation)

2. 로봇 소프트웨어를 위한 도메인 사전을 정의한다.

도메인 사전은 사용자와 개발자 간의 의사소통을 위하여 도메인 내에서 사용되는 용어로서, 모바일 로봇의 개발 과정에서 사용될 수 있는 용어들을 정확하게 정의하여야 한다.

<표 2-2>는 재난 구조용 모바일 로봇에 대한 도메인 사전의 예를 나타낸다.

<표 2-2> 재난 구조용 모바일 로봇을 위한 도메인 사전의 예

명칭	타입	설명 또는 예시
원격 모니터링	프로세스	사용자가 로봇의 상태, 로봇 센서의 데이터, 비전 센서의 비디오 등을 확인한다.
원격 제어	프로세스	사용자가 로봇을 동작 및 정지, 방향 전환 등을 지시한다.
자율 주행	프로세스	로봇이 스스로 장애물을 피하면서 지정한 곳으로 움직이며, 지정한 곳에서 매니퓰레이터를 이용하여 지정한 작업을 수행한다.
사용자	역할	로봇을 동작시키고, 위험 상황에 처한 경우 수동 조작을 통하여 안전한 행동을 하게 한다.
매니퓰레이터 수동 조작	기능	사용자가 원격으로 매니퓰레이터를 조작하여 작업을 수행한다.
긴급 정지	기능	로봇이 위험한 상황에 처한 경우 긴급 정지하고 사용자의 명령을 기다린다.
....		

3. 로봇 소프트웨어를 위한 비즈니스 규칙을 정의한다.

비즈니스 규칙은 해당 시스템이 동작하는 데 있어서 반드시 지켜져야 할 규칙, 정책을 말한다. 비즈니스 규칙은 고객, 매뉴얼, 업무 지시서, 전문가 인터뷰 등을 통하여 수집된다. <표 2-3>은 재난 구조용 모바일 로봇에 대한 비즈니스 규칙의 예를 나타낸다.

<표 2-3> 재난 구조용 모바일 로봇을 위한 비즈니스 규칙의 예

번호	비즈니스 규칙	타입	출처
1	재난 구조용 모바일 로봇은 재난 상황 시에 사용자에 의하여 재난 현장에 투입되어 인명 구조, 특정 임무 수행 등의 역할을 한다.	사실	도메인 사전
2	사용자는 모니터를 통하여 로봇을 감시하여야 하며, 로봇이 위험한 상황에 빠졌을 때 수동으로 원격 제어하여야 한다.	제약	도메인 전문가와의 인터뷰
3	로봇은 스스로 목적지가 지정되면, 스스로 장애물을 감지하여 맵을 만들고, 경로를 만들어 목적지로 이동하여야 한다.	사실	도메인 사전
4	로봇은 목적지에 도착하면, 스스로 매니퓰레이터를 조작하여 특정한 작업을 수행하여야 한다.	사실	도메인 사전
5	로봇이 수행해야 할 작업이 복잡하여 원활하게 수행하기 어려운 경우, 사용자가 원격으로 매니퓰레이터를 수동 조작할 수 있다.	유추	도메인 사전
...			

2 로봇 소프트웨어를 모델링하기 위한 UML(unified modeling language) 설계 도구를 선정한다.

1. 다음과 같은 절차에 따라 UML 설계 도구를 선정한다.

(1) <표 2-4>과 같은 프로그램 설계 도구 SW를 조사한다.

(2) 각 프로그램 설계 도구 SW의 특성에 대하여 파악한다.

(3) 시스템을 설계하기에 적합한 프로그램 설계 도구 SW를 선택한다.

(4) 선정된 프로그램 설계 도구 SW의 기본적인 사용 방법에 대하여 이해한다.

<표 2-4> UML SW의 예

구분	UML SW
commercial	RATIONAL ROSE, MAGICDRAW UML, BORLAND TOGETHER, ATOVA UMODEL, RATUNAL RHAPSODY, MISCROSOFT VISIO 등
open source	VISUAL PARADIGM FOR UML COMMUNITY EDITION, PAPYRUS, STARUML, ARGOUML, MODELIO FREE EDITION 등

2. 프로그램 설계 도구 SW를 이용하여 사용 사례 다이어그램, 컴포넌트 다이어그램, 클래스 다이어그램, 활동 다이어그램, 상태 다이어그램 등의 작성 방법에 대해 이해한다.

3 로봇 소프트웨어에 대한 사용 사례 모델링을 수행한다.

1. 로봇 소프트웨어에 대한 요구 사항을 바탕으로 시나리오를 작성한다.

개발 대상 로봇 시스템에 대한 개발 기획 보고서, 소프트웨어 요구 사항 명세서, 도메인 분석 자료, 설문 조사, 전문가 면담 등을 이용하여 로봇 소프트웨어를 위한 시나리오를 작성한다.

시나리오를 작성하고 나면 가능한 시나리오들에 대한 우선순위를 결정한다. 만약 로봇 소프트웨어에서 서로 상충되는 시나리오들이 있다면, 사용자 또는 개발 요구자와의 면담을 통하여 요구 사항을 정확하게 정리한다.

<표 2-5>는 재난 구조용 모바일 로봇을 위한 시나리오의 예이다.

<표 2-5> 재난 구조용 모바일 로봇을 위한 시나리오의 예

번호	시나리오
1	로봇의 투입 시나리오 1. 재난 현장에 사용자가 도착하면 로봇을 작동시킨다. 2. 로봇을 작동시킨 후 원격 조종기를 통하여 로봇이 수행해야 할 업무를 지시한다. 3. 로봇은 지시받은 대로 업무를 수행한다.
2	로봇의 장애물 감지 및 회피 시나리오 1. 로봇이 자율 주행 도중 로봇의 몸체에 부착된 초음파 센서를 이용하여 장애물을 감지한다. 2. 액추에이터를 제어하여 로봇을 정지시킨다. 3. 로봇이 가진 맵을 갱신한다. 4. 경로 계획을 재수행하여 목적지까지 가기 위한 경로를 계산한다. 5. 액추에이터와 스티어링을 제어하여 주행을 시작한다.
3	…….

2. 액터를 식별한다.

액터는 외부에서 시스템에 접근할 수 있는 사람이나 시스템과 관련된 외부 시스템을 의미한다. 모바일 로봇의 경우, 로봇 혼자 움직일 때는 사용자만 액터로 간주될 것이다. 만약 2대 이상의 모바일 로봇이 협력을 해야 하는 경우에는 사용자와 협력 로봇이 액터로 간주될 것이며, 모바일 로봇이 사람과 협력해야 하는 경우에는 사용자와 보조자가 액터로 간주될 것이다. 반면, 산업용 로봇의 경우 사용자, 로봇, 기계, 관리 시스템 등 공장 내에서 로봇과 상호 관계를 맺는 다양한 형태의 기기들이 될 수 있을 것이다.

[그림 2-8]은 재난 구조용 모바일 로봇이 단독 작업을 하는 경우의 액터의 예를 나타내고 있다.

[그림 2-8] 재난 구조용 모바일 로봇을 위한 액터

3. 사용 사례를 작성한다.

사용 사례는 로봇 시스템을 위한 소프트웨어의 기능을 의미한다. 즉 로봇 시스템이 수행해야 하는 일련의 행위들을 말한다. 이는 시스템에서 제공해야 하는 독립적인 기능을 의미한다. 또 외부 시스템과 상호 작용하는 행위들의 기능만을 의미한다.

따라서 사용 사례는 행위 자체만을 표현할 뿐이며, 행위 과정은 기술할 필요가 없다. 행동에

관한 구체적인 수행 방법은 사용 사례에서 표현하지 않으며 개발 과정에서 알고리즘으로 구현되면 된다.

<표 2-6>과 <표 2-7>은 재난 구조용 모바일 로봇을 위한 사용 사례의 예를 나타낸다. <표 2-6>의 로봇 시동 사용 사례의 경우 사용자가 액터로서 참여하는 경우로서, 로봇 시동 이후에 사용자에 의하여 로봇 작업 지시가 이루어진다. 즉, 로봇 시동과 로봇 작업 지시는 <<extend>> 관계에 있다. 반면, <표 2-7>의 장애물 회피 사용 사례의 경우 액터가 없는 경우로서, 장애물 감지 사용 사례가 수행된 후에 수행되는 사용 사례이다. 즉, 장애물 감지와 장애물 회피 사용 사례는 <<include>> 관계에 있다.

<표 2-6> 재난 구조용 모바일 로봇의 사용 사례: 로봇 시동

사용 사례 이름	로봇 시동
참여 액터	사용자
시작 조건	사용자가 시동 버튼을 누른다.
사건의 흐름	- 정상 흐름 1. 초기화 루틴을 수행하여 각 기능들이 정상적으로 동작하는지를 체크한다. 2. 정상 동작이 확인되면, 정상 동작을 알리는 LED를 on시킨다. 3. 사용자로부터 명령어를 입력받을 수 있는 상태에서 대기한다. - 선택 흐름 1. 정상 동작이 확인되지 않으면, 에러 사항을 알리는 LED를 on시킨다. 2. 사용자가 유선 또는 무선으로 에러 코드를 받을 수 있도록 대기한다.

<표 2-7> 재난 구조용 모바일 로봇의 사용 사례: 장애물 회피

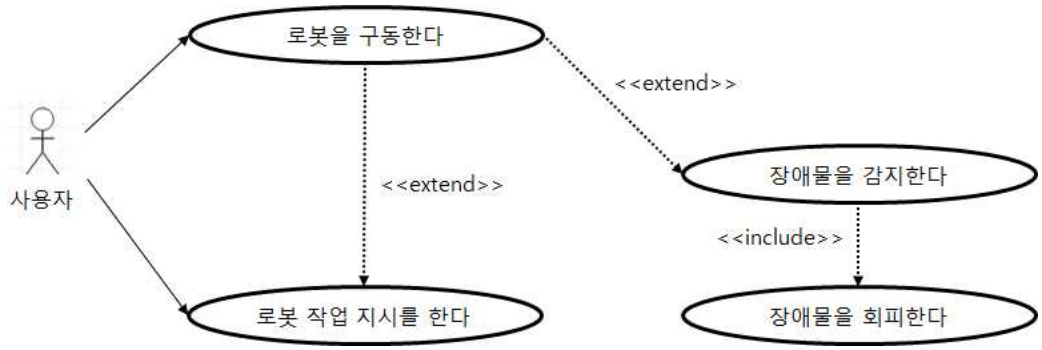
사용 사례 이름	장애물 회피
참여 액터	없음
시작 조건	초음파 센서 동작 사용 사례에서 장애물이 감지된다.
사건의 흐름	- 정상 흐름 1. 액추에이터를 제어하여 로봇을 정지시킨다. 2. 감지된 장애물을 맵에 표시하고, 맵을 갱신한다. 3. 경로 계획을 재수행하여, 목적지까지 가기 위한 최적의 경로를 계산한다. 4. 액추에이터와 스티어링을 제어한다. - 선택 흐름 1. 경로 계획 작업 결과 경로를 찾지 못했을 경우, 사용자에게 경로를 찾지 못하였다는 메시지를 보낸다. 2. 사용자는 원격 제어기를 이용하여 수동으로 모바일 로봇을 조정한다.

4. UML 설계 도구를 이용하여 사용 사례 다이어그램을 도식화한다.

선정된 액터와 사용 사례를 가지고 사용 사례 다이어그램을 작성한다.

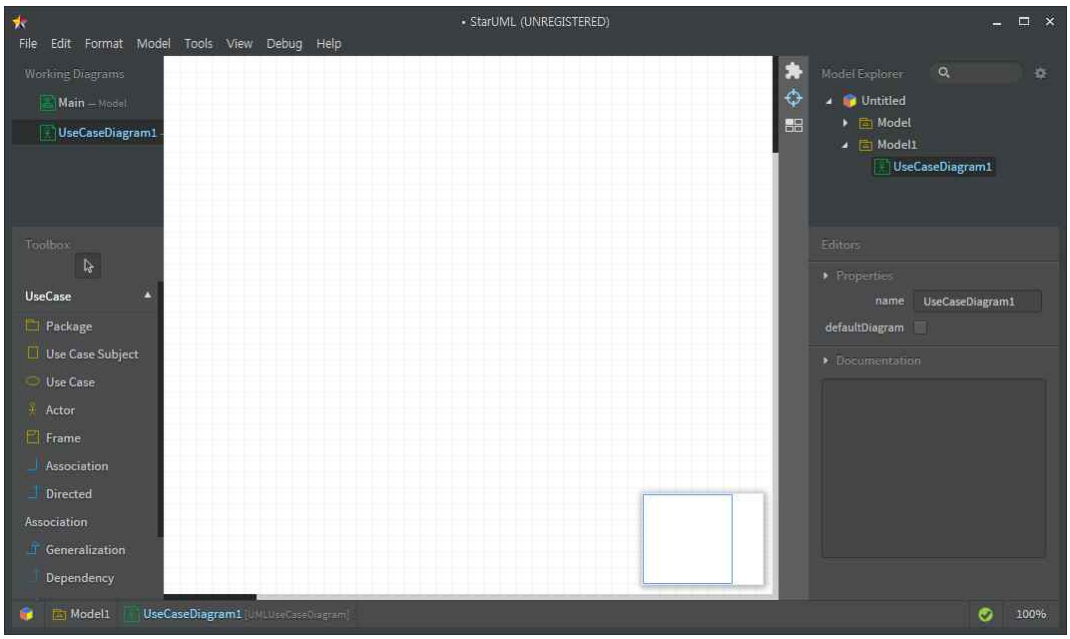
[그림 2-9]는 재난 구조용 모바일 로봇을 위한 사용 사례 다이어그램의 예를 나타낸다. 아래 그림에서는 사용 사례로 로봇 시동, 로봇 작업 지시, 장애물 감지, 장애물 회피와 같은 4개의 사용 사례만을 이용하여 다이어그램을 작성하였다.

실제로 로봇 시스템이 개발될 때에는 훨씬 더 많은 사용 사례가 있을 것이며, 각 사용 사례와의 관계를 명확하게 표현하여야 할 것이다.



[그림 2-9] 재난 구조용 모바일 로봇을 위한 사용 사례의 예

사용 사례 다이어그램은 UML 설계 도구를 이용하여 그릴 수 있다. [그림 2-10]은 무료로 사용이 가능한 StarUML을 나타낸다. StarUML 소프트웨어는 GNU 공개 라이센스로 오픈 소스 프로그램으로서, 홈페이지(http://staruml.io/)에서 다운로드 가능하다.



출처: StarUML(http://www.staruml.io). 2016. 9. 30. 스크린샷.

[그림 2-10] StarUML 소프트웨어의 실행

수행 tip

- UML 모델링 도구로 다양한 제품들이 나와 있다. UML 다이어그램 제작을 위하여 관련 소프트웨어에 대한 이해가 선행되어야 한다.

학습 1	로봇 소프트웨어 요구 사항 정의하기
학습 2	로봇 소프트웨어 사용 사례 작성하기
학습 3	**로봇 소프트웨어 아키텍처 정의하기**
학습 4	로봇 소프트웨어 아키텍처 설계하기

3-1. 로봇 소프트웨어 아키텍처 정의

학습 목표

- 로봇 요구 사항 분석에 따라 요구되는 주요 소프트웨어 항목을 분석할 수 있다.
- 주요 소프트웨어 항목을 구성하기 위한 소프트웨어 아키텍처의 요구 사항을 도출할 수 있다.
- 소프트웨어 아키텍처의 요구 사항에 근거하는 소프트웨어 개발 환경을 결정할 수 있다.
- 로봇 요구 사항에 따르는 로봇 소프트웨어 아키텍처를 계층적 구성 방식으로 분류하여 정의할 수 있다.
- 로봇 소프트웨어 아키텍처를 구성하는 다양한 소프트웨어를 주요 기능별로 통합하거나 분리하여 모듈화할 수 있다.
- 단위 모듈의 역할을 담당하는 소프트웨어 컴포넌트의 기능을 정의할 수 있다.
- 소프트웨어 컴포넌트 간 통신 방식과 프로토콜을 정의할 수 있다.

필요 지식 /

[1] 소프트웨어 아키텍처

소프트웨어 아키텍처는 소프트웨어를 구성하는 컴포넌트와 컴포넌트의 관계를 추상적인 수준에서 정의하는 것을 말한다. 즉, 소프트웨어 전체 구조를 한눈에 볼 수 있다면, 각 컴포넌트와 컴포넌트 사이의 관계에 대한 근거를 판단할 수 있다. 또 소프트웨어 아키텍처는 소프트웨어에 대한 이해를 돕고 시스템 수준의 설계 모델을 재사용할 수 있게 해주며, 품질 요구 사항을 반영할 수 있게 해 주며, 소프트웨어 상세 설계 및 구현 이전 초기 설계 단계에서 소프트웨어가 가질 품질 요구 사항을 예측할 수 있게 해준다.

예를 들면 휴머노이드 로봇 시스템은 여러 개의 액추에이터를 제어해야 하는 액추에이터 제어 관련 소프트웨어, 비전 센서와 같은 센서 입력을 다루는 다양한 형태의 센서 관련

소프트웨어, 휴머노이드 로봇을 움직이기 위하여 여러 개의 액추에이터를 통합 제어해야 하는 모션 컨트롤 관련 소프트웨어, 사람들을 피해서 특정 지점까지 갈 수 있는 경로를 만들어 주는 경로 계획 소프트웨어 등과 같은 다양한 소프트웨어들을 포함하고 있다.

이러한 소프트웨어들을 추상화시켜서 컴포넌트라 부른다. 컴포넌트는 명백한 역할을 가지고 있으며 독립적으로 존재할 수 있는 시스템의 부분으로 정의된다. 예로 휴머노이드 로봇 시스템에서 비전 센서나 모션 컨트롤 등과 같은 소프트웨어는 로봇 시스템의 서브시스템으로서 컴포넌트로 정의될 수 있다.

이러한 컴포넌트들은 재사용 가능하도록 설계되어야 한다. 즉, 현재 사용되고 있는 모션 컨트롤러를 다른 컨트롤러로 변경할 경우, 변경된 컨트롤러에 관한 소프트웨어만 변경하면 나머지 다른 기능들, 즉 비전 센서 관련 소프트웨어의 변경 없이도 로봇 시스템이 정상적인 동작을 할 수 있다는 의미이다. 이러한 재사용성을 만족하려면, 컴포넌트들 간의 인터페이스와 상호 작용에 대하여 명확히 정의가 되어 있어야 한다.

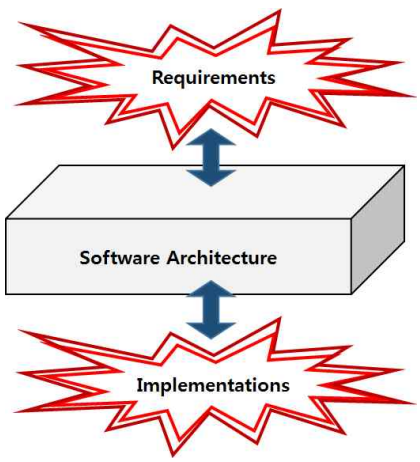
일반적으로 소프트웨어 아키텍처가 중요한 이유는 다음과 같다.

첫째, 소프트웨어 아키텍처는 이해 관계자와의 커뮤니케이션을 위하여 정의된 형상이자 표현의 수단이다. 소프트웨어 아키텍처를 통해 이해 관계자들(프로젝트 관리자, 품질 담당자, 개발자, 테스터, 고객 등)은 보다 원활한 커뮤니케이션을 할 수 있다.

둘째, 소프트웨어 아키텍처는 설계 방향의 가이드가 된다. 전체적인 관점에서 품질 속성을 고려하여 목표 시스템의 설계 방향을 조기에 결정할 수 있도록 지원하고, 향후 발생할 수 있는 위험 요소를 감소시킬 수 있다.

셋째, 소프트웨어 아키텍처는 재사용 가능한 시스템의 추상화를 제공함으로써 소프트웨어 아키텍처가 시스템을 구성하는 요소와 이들 간의 관계를 간략하고 명확하게 나타낼 수 있다. 따라서 비슷한 기능과 품질 요소를 가지는 다음 시스템에서의 큰 규모의 재사용이 가능하고, 소프트웨어 조직의 자산으로써 재사용 또한 용이해진다.

소프트웨어 아키텍처는 [그림 3-1]과 같이 요구 사항과 구현 사이에서, 요구 사항을 제대로 구현하기 위해 도움을 줄 수 있는 커뮤니케이션 툴이자 개발 초기의 요구 사항과 실제 솔루션 간의 간극을 좁혀줄 수 있는 훌륭한 도구이기도 하다.



[그림 3-1] 소프트웨어 아키텍처의 역할

2 로봇 소프트웨어 아키텍처

1. 로봇 소프트웨어 프레임워크

로봇 시스템에서는 로봇 소프트웨어 프레임워크(framework)라는 용어를 사용하기도 한다. 일반적으로 소프트웨어 공학에 있어서 프레임워크(framework)는 API(application programming interface)와 같은 라이브러리로 잘 정의되어서 재사용이 가능한 소프트웨어이다. 자바의 JDK나 마이크로소프트의 .NET framework가 프레임워크의 좋은 예라고 볼 수 있다.

그러나 로봇 시스템에서 프레임워크는 조금 다른 의미로 해석된다. 즉, 로봇 시스템에서는 재사용이 가능한 소프트웨어라는 점에서는 소프트웨어 라이브러리와 유사하나, 전체적인 프로그램의 제어 흐름이 사용자에 의하여 지배되는 것이 아니라 프레임워크에 의하여 결정된다는 점에서 큰 차이를 가지고 있다.

이러한 로봇 시스템 프레임워크, 또는 프레임워크와 개발 환경이 통합되어 있는 로봇 플랫폼(platform)은 로봇 시스템이 여타의 다른 소프트웨어와는 다른 특수성을 가지고 있기 때문이다. 산업 현장에서 로봇 시스템을 제작할 경우 모터나 센서, 모션 컨트롤러 등과 같은 부품들을 기성품으로 구매한 후 이를 통합하는 작업, 즉 로봇 소프트웨어 개발과 상용화 작업을 거친다. 따라서 로봇 소프트웨어 개발자에게 상용화되어 있는 부품들의 소프트웨어를 쉽게 사용할 수 있다면 시스템 통합 작업이 단순해지고 프로젝트의 개발 시간을 줄여줄 수 있을 것이다.

이에 대한 해결책이 로봇 소프트웨어 플랫폼이다. 로봇 소프트웨어 플랫폼이란 로봇 응용 프로그램을 개발할 때 필요한 하드웨어 추상화, 하위 디바이스 제어, 로보틱스에서 많이 사용되는 센싱, 인식, 자기 위치 추정과 지도 작성(SLAM), 모션 플래닝(motion planning) 등의 기능 구현은 물론이고, 패키지 관리, 개발 환경에 필요한 라이브러리와 다양한 개발/디버깅 도구 등을 포함하는 것을 말한다.

소프트웨어 플랫폼에는 대표적으로 로봇 운영체제라 불리는 OPEN SOURCE ROBOTICS FOUNDATION의 ROS(robot operating system), 일본 AIST의 OpenRTM, 유럽의 OROCOS, 한국의 OPRoS, 마이크로소프트의 MSRDS, EVOLUTION ROBOTICS의 ERSP 등이 있다.

2. 로봇 소프트웨어 아키텍처의 계층 구조

[그림 3-2]는 로봇 소프트웨어 아키텍처의 계층 구조를 나타내고 있다.

(1) 하드웨어 계층

최하위 계층은 하드웨어 계층으로서, 액추에이터와 센서를 구동하기 위한 드라이버들이 정의되어 있다.

(2) 로봇 실시간 운영체제 계층

다음 계층은 운영체제(OS, operating system) 계층으로서 하드웨어의 관리와 어플리케

이션의 관리, 그리고 어플리케이션이 하드웨어를 활용할 수 있도록 인터페이스를 제공하는 일련의 소프트웨어의 집합이다. 여기서 관리의 의미는 하드웨어 접근의 권한과 제한된 하드웨어를 어플리케이션들이 어떻게 공유하는지에 대한 방법들을 의미한다.

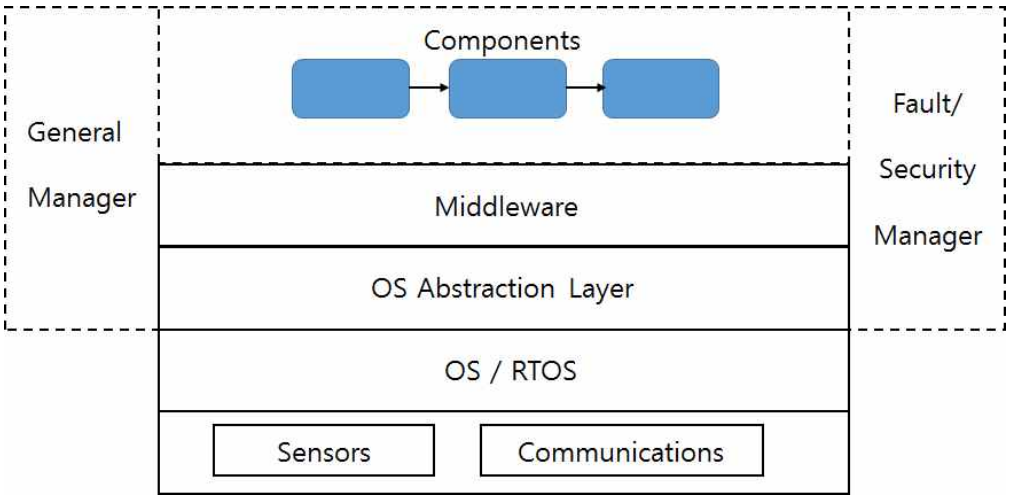
로봇 시스템에서는 실시간 운영체제(RTOS, real-time operating system)를 주로 사용한다. 실시간 운영체제는 일반 운영체제에 비하여 시분할의 방법 및 정밀도가 우수하며, 정밀 제어를 위해 폭넓게 사용된다. 즉, 프로그램 내에서의 작업을 나눈 태스크(task)가 다수 존재하는 경우 이들의 우선순위와 실행되는 빈도를 정밀하게 조정할 수 있다. 예로 로봇의 경우 센서를 처리하는 태스트는 10 msec마다 실행해야 되고 모터를 제어하는 태스트는 5 ms마다 실행해야 된다면, 실시간 운영체제는 센서와 모터에 관한 태스크들이 주기적으로 제 시간에 동작되도록 태스크들을 관리한다.

(3) 로봇 미들웨어 계층

로봇 미들웨어 계층은 로봇의 전체적인 운영은 물론 각종 모듈의 제어를 위한 라이브러리와 관리 프로그램을 포함하는 것으로 운영체제 위에서 동작하는 응용 계층을 지원하는 플랫폼이라고 할 수 있다. 로봇의 복잡도가 증가함에 따라 산업용 로봇은 물론 서비스용 로봇에 이르기까지 광범위하게 적용되고 있다.

(4) 로봇 응용 프로그램 계층

로봇 응용 프로그램 계층은 미들웨어에서 API로 만들어져 있는 라이브러리를 이용하여 작성되고 운영 체제의 관리 하에서 실행된다.



[그림 3-2] 로봇 소프트웨어 아키텍처의 계층 구조

3 컴포넌트 다이어그램

1. 컴포넌트 다이어그램

컴포넌트 다이어그램에서 컴포넌트는 시스템을 구성하는 임의의 물리적인 요소를 의미한다. 물리적인 요소란 가상의 모델을 실제로 구현하여 나타내는 것을 의미한다.

컴포넌트는 객체 지향 프로그램의 원리에 따라 업무 기능과 관련 데이터를 하나의 단위로 처리하고 있다. 즉, 컴포넌트란 인터페이스에 의해서 기능이 정의된, 독립적으로 개발·배포·조립이 가능한 시스템의 구성단위이다. 로봇 소프트웨어 아키텍처에서 경로 계획 등과 같은 모듈들은 컴포넌트의 예가 된다.

2. 컴포넌트 다이어그램의 구성 요소

컴포넌트 다이어그램은 시스템을 구성하는 물리적인 컴포넌트와 그들 사이의 의존 관계를 나타내는 다이어그램이다. 컴포넌트 다이어그램은 컴포넌트와 인터페이스로 표현된다.

(1) 컴포넌트

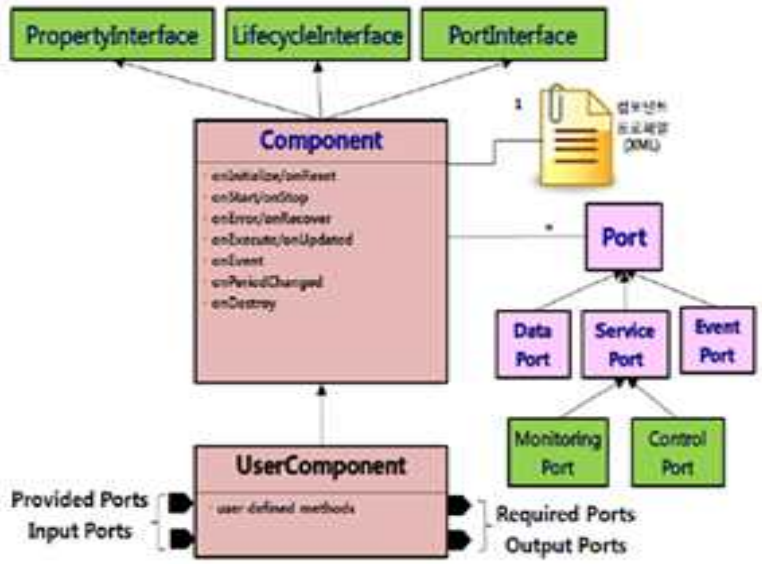
컴포넌트는 탭이 달린 직사각형으로 표현되며, 모든 컴포넌트는 반드시 이름을 가지고 있어야 한다.

[그림 3-3]은 OPRoS 소프트웨어 아키텍처에서 제공하는 컴포넌트를 나타내고 있다. OPRoS에서 컴포넌트는 컴포넌트의 특성을 기술하는 컴포넌트 프로파일(component profile)을 가지고 있다. 컴포넌트들의 조합을 용이하게 하기 위해 컴포넌트의 특성 및 외부 인터페이스 등에 대한 정보를 기술하는 컴포넌트 프로파일 정보가 컴포넌트와 함께 제공된다.

컴포넌트는 하나 이상의 포트로 구성되며, 포트는 종류에 따라 서비스 포트, 데이터 포트, 이벤트 포트로 나뉘며 기본 포트(port)를 상속하여 만들어진다. 각 컴포넌트는 컴포넌트의 생명 주기 관리를 위한 제어 서비스 포트(control service port)와 모니터링을 위한 모니터링 서비스 포트(monitoring service port)를 디폴트로 가지고 있다.

제어 서비스 포트를 이용하여 컴포넌트 외부에서 해당 컴포넌트의 초기화, 시작, 중지, 일시 정지, 재시작 등의 생명 주기 관련 API를 호출할 수 있으며, 모니터링 서비스 포트를 통해 해당 컴포넌트의 상태를 모니터링할 수 있다. 디폴트 포트 이외에 사용자가 정의하는 포트는 portinterface의 API를 통해 컴포넌트에 추가할 수 있다.

사용자가 개발하려고 하는 유저 컴포넌트는 OPRoS 컴포넌트에서 제공하는 추상 컴포넌트인 component를 상속 받아 구현해야 한다. component는 property 관리를 위한 propertyinterface와 생명주기 관리를 위한 lifecycleinterface, 그리고 포트 관리를 위한 portinterface를 상속받은 추상 컴포넌트이다. 컴포넌트의 구현에서는 사용자는 component를 상속받아 필요한 콜백 함수를 정의하고, 사용자 정의 메소드를 추가하기만 하면 된다.



출처: OPRoS 튜토리얼(http://ropros.org). 2016. 9. 30. 스크린샷.
[그림 3-3] OPRoS의 컴포넌트 클래스

(2) 인터페이스

인터페이스는 컴포넌트와 컴포넌트 간의 관계를 표현하기 위하여 사용된다.

[그림 3-4]는 OPRoS 소프트웨어 아키텍처에서 외부 컴포넌트와의 인터페이스 방법을 나타내고 있다. OPRoS에서 컴포넌트에서는 외부 컴포넌트와의 인터페이스를 포트(port)를 통해 수행한다. OPRoS에서 포트 종류에는 서비스 포트(service port), 데이터 포트(data port)와 이벤트 포트(event port)가 있다. 외부에 서비스를 제공하는 서비스 포트를 provided 포트라고 하며, 외부의 서비스를 이용해야하는 경우에는 required 포트라고 한다.

또 데이터 포트 및 이벤트 포트는 외부로부터 데이터(혹은 이벤트)를 수신하는 포트를 input 포트라고 하며, 외부에 데이터(혹은 이벤트)를 송신하는 포트를 output 포트라고 한다.

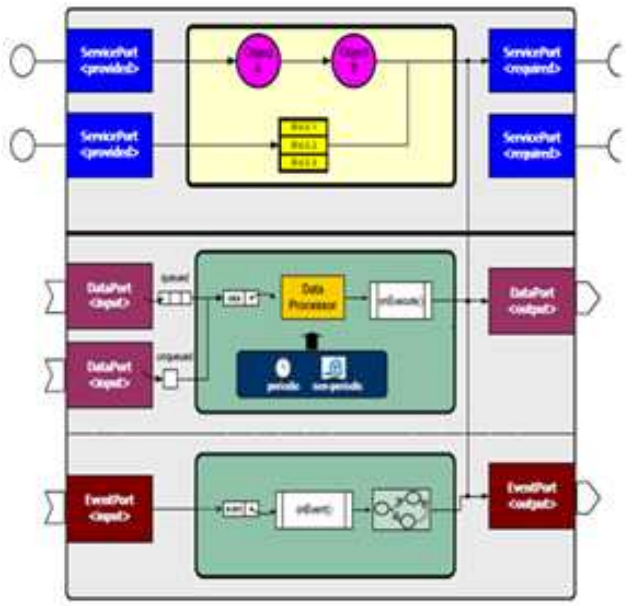
컴포넌트는 서비스 포트, 데이터 포트, 이벤트 포트 중 적어도 하나의 포트를 갖고 있어서 외부와의 인터페이스를 수행한다. 또 컴포넌트는 용도에 따라 다양하게 포트를 생성하여 사용할 수 있으며, 같은 종류의 포트를 여러 개 생성하여 컴포넌트를 구성할 수 있다.

컴포넌트는 서비스 포트를 통해 다른 컴포넌트가 제공하는 메소드를 호출하거나 해당 컴포넌트의 속성에 접근할 수 있으며, 데이터/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/이벤트를 전달하기 위해서는 해당 컴포넌트의 포트를 알아야 한다. 이를 위해 컴포넌트 개발자는 해당 컴포넌트가

제공하는 포트와 다른 컴포넌트가 사용할 수 있는 포트를 인터페이스 프로파일(interface profile)에 명시해야 한다.

서비스 포트는 컴포넌트 내부의 메소드들과 바인딩되어 사용되며, 데이터 포트는 데이터 처리기(data processor)와 바인딩되어 주기적인(periodic) 방식이나 비주기적(non-periodic) 방식으로 컴포넌트의 onExecute( )를 호출하여 데이터를 처리한다. 이벤트 포트는 이벤트가 도착 시에 특정 메소드(onEvent( ))가 즉시 호출되어 상태 관리를 하도록 한다.



출처: OPRoS 튜토리얼(http://ropros.org). 2016. 9. 30. 스크린샷.
[그림 3-4] OPRoS의 컴포넌트 클래스

수행 내용 / 소프트웨어 아키텍처 정의하기

재료・자료

- 로봇 하드웨어 사양서
- 로봇 운영체제 사양서
- 로봇 기능 정의서
- 로봇 소프트웨어 프레임워크
- 로봇 소프트웨어 아키텍처 요구 사항 정의서

- 소프트웨어 아키텍처 접근법
- 소프트웨어 아키텍처 설계 지침서
- 오픈소스 지적 재산권 지침서

기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 로봇 소프트웨어 프레임워크 개발 환경툴(SDK)
- 프로그램 개발용 소프트웨어
- 컴파일러
- 문서 작성용 소프트웨어
- 소프트웨어 품질 검사 도구

안전 · 유의 사항

- 로봇의 형태에 따른 작업 능력에 대한 정확한 숙지와 특성 파악이 필요하다.
- 로봇 소프트웨어 프레임워크 이용 시에는 저작권 관련 규정을 준수하여야 한다.

수행 순서

[1] 소프트웨어 아키텍처 설계 원리 및 설계 과정에 대해 이해한다.

1. 소프트웨어 아키텍처 설계 원리에 대해 이해한다.

   소프트웨어 아키텍처 설계에서 중심이 되는 원리인 단계적 분해(stepwise refinement), 추상화(abstraction), 모듈화(modularization)에 대해 이해한다.

   (1) 단계적 분해에 대해 이해한다.

   모바일 로봇과 같은 로봇 소프트웨어는 매우 복잡한 소프트웨어를 가지고 있기에, 하나의 덩어리로 다루는 것은 매우 어렵다. 따라서 소프트웨어 설계에서의 목표는 로봇 소프트웨어를 다루기 쉽고 풀 수 있는 작은 조각으로 나누는 것이다.

   분해를 할 때에는 각 부분을 별도로 독립적으로 해결할 수 있는지를 확인해야 한다. 분해를 하게 되면 분해된 조각들이 문제를 해결하기 위하여 협력하고 소통하여야 한다. 이를 인터페이스라고 하는데, 이러한 인터페이스는 시스템의 복잡도를 증가시키는 요인이 된다. 따라서 소프트웨어를 분해할 때에는 개별 문제를 해결할 수 있는 조각들의 비용의 합이 문제를 전체로 다루어 해결하려는 비용보다 작아야 한다.

단계적 분해를 위한 가장 중요한 기준 중의 하나는 단순성과 이해성이다. 로봇 시스템의 각 부분이 애플리케이션과 단순하게 관련되고 각각 독립적으로 수정될 수 있다면 유지보수나 시스템 업그레이드, 시스템 통합 등에 들어가는 노력이 최소화될 수 있다. 또 적절하게 분해된 각 부분들은 이해하기 쉬워지기 때문에 시스템 통합을 용이하게 한다.

예를 들어 [그림 1-3]에서 언급된 매니퓰레이터가 탑재된 모바일 로봇의 경우, 모바일 로봇을 위한 주행 소프트웨어와 매니퓰레이터를 위한 동작 소프트웨어는 서로 독립적으로 동작한다. 그러나 매니퓰레이터가 특정 지점까지 로봇 팔을 뻗지 못할 경우 모바일 로봇을 움직여야 될 수도 있다. 이러한 동작에 관한 정보를 인터페이스로 정의해야 한다. 따라서 두 소프트웨어는 인터페이스에 관한 정보만 입출력으로 공유할 수 있다면 완전히 독립적으로 제작될 수 있다.

(2) 추상화에 대해 이해한다.

추상화는 서브시스템을 구현하는 방법에 대해서는 관심을 가지지 않고, 서브시스템이 외부에 어떻게 서비스를 제공하는지에 대해서만 관심을 가지자는 것이다.

위의 예에서 모바일 로봇의 주행을 위해서는 캐터필터에 연결된 모터를 제어해야 하며 차량 동역학에 맞추어 방향 전환 등을 수행해야 한다. 이를 위해서 주행 소프트웨어에는 모터 제어 알고리즘과 주행 알고리즘이 포함되어 있어야 한다. 반면 매니퓰레이터의 동작을 위해서는 각 관절에 있는 모터를 제어해야 하고 역기구학을 이용하여 특정 좌표로 로봇 팔을 이동시켜야 한다. 이를 위해서 매니퓰레이터 동작 소프트웨어는 모터 제어 알고리즘과 로봇 팔 제어 알고리즘이 포함되어 있어야 한다.

추상화를 할 때에는 이러한 내부 알고리즘에 대해서는 고려하지 않고 2개의 소프트웨어가 어떤 상호 작용을 하는지에 대해서만 초점을 맞춘다. 즉 모바일 로봇의 주행 소프트웨어는 목적지까지의 좌표만 입력받으면 내부의 여러 가지 알고리즘을 이용하여 목적지까지 주행을 하게 될 것이다. 반면, 매니퓰레이터의 동작 소프트웨어는 로봇 팔이 가야되는 지점의 좌표를 입력받으면 해당 작업을 수행하게 될 것이다.

(3) 모듈화에 대해 이해한다.

시스템을 모듈로 분할하면 각각의 모듈을 별개로 만들고 수정할 수 있다. 또 모듈을 별도로 컴파일할 수 있다면 모듈을 수정해도 전체 시스템을 다시 컴파일할 필요가 없다. 따라서 시스템을 구별된 모듈들로 나누고 각 컴포넌트를 별도로 구현할 수 있다면 모듈화되었다고 간주한다.

위의 예에서 보듯이 주행 모듈과 매니퓰레이터 동작 모듈의 경우 별개로 만들 수 있고, 별도로 컴파일이 가능하다. 이렇게 모듈화를 잘 시키면 로봇 시스템 통합이 용이해질 것이다.

2. 소프트웨어 아키텍처 설계 과정에 대해 이해한다.

소프트웨어 아키텍처 설계 과정은 다음과 같으며, 부록에 첨부한 소프트웨어 아키텍처 정의서의 양식과 유사한 방법으로 작성한다.

(1) 제안서, 프로젝트 계획서, 인터뷰 자료, 요구 사항을 파악한다.

(2) 시스템을 구축하기 위한 기술적인 요소가 무엇인지 파악한다.

(3) 시스템의 제약 사항은 무엇인지 파악한다.

(4) 위의 항목을 정리하여 시스템이 다른 시스템과 구별되는 특성은 무엇인지 서술한다.

(5) 요구 사항과 유스케이스를 바탕으로 시스템에 대한 아키텍처 초안을 작성한다.

(6) 아키텍처의 구성 요소 간 연결 관계를 파악한다.

(7) 아키텍처가 만족해야 하는 품질 항목은 무엇인지 파악한다.

(8) 아키텍처 초안을 구체화하고 추가적인 컴포넌트와 연결 관계를 파악한다.

(9) 아키텍처를 리뷰하고 프로젝트에서 적절한지 판단한다.

(10) 기술적인 의사 결정 항목을 나열한다.

(11) 의사 결정 항목의 각 항목에 대해 서술한다.

(12) 설계 및 구현에 대한 가이드라인을 작성한다.

(13) 설계와 구현이 가능하도록 설계 예제와 구현 예제를 작성한다.

2 로봇 소프트웨어 아키텍처의 계층 구조 및 각 계층에서 다루는 기능에 대해 이해한다.

1. 하드웨어 계층에서 다루는 기능에 대해 이해한다.

(1) MCU(micro controller unit)의 펌웨어에 대해 이해한다.

간단한 로봇은 MCU 상에서 펌웨어(firmware)로 모든 제어를 하는 경우가 있다. 또 서보 드라이브 소프트웨어나 센서 소프트웨어의 경우 MCU 상에서 펌웨어로 제어가 되는 경우가 많다.

로봇 MCU 및 하드웨어 설계 및 제작 학습모듈을 참조하여 라인 트레이스와 같은 간단한 로봇을 대상으로 펌웨어 프로그램을 작성하는 방법에 대해 이해한다.

(2) 모터 드라이브 소프트웨어에 대해 이해한다.

모터를 제어하기 위하여 모터의 전압 및 전류를 제어하는 장치에 들어가는 서보 드라이브를 위한 소프트웨어로서, 상위 제어기에 있는 모션 컨트롤 소프트웨어로부터 지령을 받아 모터의 회전 각도와 속도를 제어한다.

서보 드라이브 소프트웨어에 대하여 로봇 액추에이터 드라이버 설계와 로봇 액추에이터 제어 소프트웨어 개발 학습모듈을 참조하고, 상위 제어기에 대하여 로봇 모션 제어

기 하드웨어 설계와 로봇 모션 제어 소프트웨어 개발 학습모듈을 참조하여 해당 사항에 대해 이해한다.

(3) 센서 소프트웨어에 대해 이해한다.

사람의 눈 역할을 하는 카메라의 이미지를 처리하는 알고리즘을 가진 센서 소프트웨어나, 초음파 센서로부터 노이즈를 제거하는 필터 등을 가진 센서 소프트웨어 등이 있다. 센서 소프트웨어 대하여 로봇 센서 신호처리부와 로봇 센서 인터페이스 설계 학습모듈을 참조하여 해당 사항에 대해 이해한다.

2. 실시간 운영체제에 대해 이해한다.

실시간 운영체제는 로봇을 비롯하여 산업 제어 시스템, 통신 장비 및 국방 장비 등 광범위한 분야에서 적용되고 있다. 실시간 운영체제는 일반 운영체제 기반 실시간 운영체제와 전용 실시간 운영체제로 나눌 수 있다.

일반 운영체제 기반 실시간 운영체제로는 WINDOWS나 LINUX와 같은 범용 운영체제에서 운영체제 일부를 바꾸거나 미들웨어 형식으로 실시간 기능을 추가한 시스템이다. 윈도우즈에서는 RTX라는 시스템이 주로 사용되고 있으며, 리눅스에서는 RTAI라는 추가적 시스템이 많이 사용되고 있다.

전용 실시간 운영체제는 실시간 기능을 중점으로 만들어진 운영체제로서 수많은 제품이 존재한다. 현재 주로 많이 사용되고 있는 것은 VxWorks와 uCOS 등이 있다. VxWorks는 유료로서 고가의 장비에 주로 사용되며, uCOS는 무료로서 소형의 시스템이나 학생들의 교육에 주로 활용된다.

실시간 운영체제에 대해서는 로봇 미들웨어 설계 학습모듈을 참조하여 해당 사항에 대해 이해한다.

3. 로봇 미들웨어에 대해 이해한다.

로봇 미들웨어는 실제 로봇의 동작 계산을 위한 기구학(kinematics) 및 역기구학(inverse kinematics), 동역학(dynamics), 모션 컨트롤, 각종 센서의 입력을 이용하여 특정 목적을 수행하는 알고리즘 등을 컴포넌트화하여 라이브러리 형태로 제공한다. 이들 라이브러리는 운영 체제의 API를 이용하여 제작되며 운영 체제의 특성을 효율적으로 이용하여 로봇 응용 프로그램의 개발에 있어 보다 효율적인 방법을 제공한다.

로봇 미들웨어로는 OPEN SOURCE ROBOTICS FOUNDATION의 ROS(robot operating system), 일본 AIST의 OpenRTM, 유럽의 OROCOS, 한국의 OPRoS, 마이크로소프트의 MSRDS, EVOLUTION ROBOTICS의 ERSP 등이 있다.

로봇 미들웨어에 대해서는 로봇 미들웨어 설계 학습모듈을 참조하여 해당 사항에 대해 이해한다.

4. 로봇 응용 프로그램에 대해 이해한다.

로봇 응용 프로그램은 일반 컴퓨터에서와 마찬가지로 MCU가 실행할 수 있는 머신 코드로 컴파일이 되어야 한다. 컴파일러는 로봇 미들웨어서 제공하는 컴파일러를 이용한다.

로봇 응용 프로그램에 대해서는 로봇 UX UI 개발, 로봇 지능 소프트웨어 개발, 로봇 콘텐츠 소프트웨어 개발, 로봇 운영 및 통합 소프트웨어 개발 학습모듈을 참조하여 해당 사항에 대해 이해한다.

3 모바일 로봇을 위한 로봇 소프트웨어 아키텍처를 정의해본다.

[그림 3-5]는 매니퓰레이터를 가진 모바일 로봇을 위한 로봇 소프트웨어 아키텍처를 나타내고 있다. 모바일 로봇을 위한 로봇 소프트웨어는 하드웨어 계층, 미들웨어 계층, 응용 계층에 관한 컴포넌트 즉 라이브러리를 가지고 있다. 그리고 일부 형태의 사용자 연동이 있는 로봇 시스템에서는 사용자 인터페이스 계층이 포함될 수도 있다.

그림과 같은 아키텍처는 경로 계획, 장애물 회피, 맵핑(mapping)을 포함한 태스크를 수행하기 위해 제작된 매니퓰레이터가 있는 모바일 로봇을 나타낸다. 이런 유형의 로봇은 탐색 및 구조와 같은 여러 가지 환경에서 사용될 수 있다. 로봇 소프트웨어들을 4개의 계층으로 나누면 다음과 같다.

1. 하드웨어 계층의 컴포넌트에 대해 정의한다.

하드웨어 계층에는 각종 센서 인터페이스와 액추에이터 인터페이스 관련 함수를 정의되어야 한다.

첫째, 센서 인터페이스 컴포넌트(sensor interface component)에는 아날로그 센서 신호를 디지털 값으로 변환하는 ADC(analog to digital converter)와 로우 레벨 신호에서 노이즈를 제거하는 소프트웨어 필터가 구현되어야 한다.

둘째, 액추에이터 인터페이스 컴포넌트(actuator interface component)에는 모터 제어를 위한 전압 제어와 일정한 속도 또는 토크를 유지하기 위한 제어 이론이 구현되어야 한다.

2. 미들웨어 계층의 컴포넌트에 대해 정의한다.

미들웨어 계층에는 다음과 같은 내용들이 정의되어야 한다.

첫째, 로봇의 조향에 관한 컴포넌트(steering component)가 정의되어야 한다. 바퀴나 캐터필더 등을 이용하여 로봇의 움직임을 제어하기 위하여 액추에이터 인터페이스 컴포넌트로로 양 바퀴의 속도를 계산하여 전달하는 컴포넌트가 정의되어야 한다.

둘째, 엔코더와 IMU(inertial measurement unit)의 데이터를 결합하여 자기 위치 인식(localization)을 수행하거나 초음파와 IR 센서, 레이저 스캐너 등을 이용하여 장애물을 감지하는 센서 퓨전 컴포넌트(sensor fusion component)가 정의되어야 한다.

셋째, 카메라를 이용하여 매니퓰레이터가 대상을 집어 올릴 수 있도록 대상을 식별할 수 있는 이미지 처리 컴포넌트(image processing component)가 정의되어야 한다.

넷째, 특정 좌표로 로봇 팔을 이동시킬 수 있는 매니퓰레이터의 기구학이나 동역학에 관한 컴포넌트(kinematics component)가 정의되어야 한다.

3. 응용 계층의 컴포넌트에 대해 정의한다.

응용 계층에는 다음과 같은 내용들이 정의되어야 한다.

모바일 로봇이 목표점까지 이동하기 위해서는 작업 공간에 대해 파악하고, 장애물을 회피하여 목표점까지 갈 수 있는 경로를 계산하여야 한다. 따라서 SLAM(simultaneous localization and mapping)과 같은 방법을 이용하여 로봇 주위의 맵을 인식할 수 있는 맵핑 및 로컬라이제이션 컴포넌트(mapping and localization component)가 정의되어야 한다.

또 장애물을 회피하기 위하여 모터 구동에 관한 명령을 내릴 수 있는 장애물 회피 컴포넌트(collision avoidance component)가 정의되어야 하며, 목표점까지 최적의 주행 경로를 생성할 수 있는 경로 계획 컴포넌트(path planning component)가 정의되어야 한다.

매니퓰레이터는 특정 지점까지 끝점을 이동시키기 위하여 각 관절의 각도에 대한 궤적 제어 또는 경로 계획 컴포넌트(trajectory control or path planning component)가 정의되어야 한다.

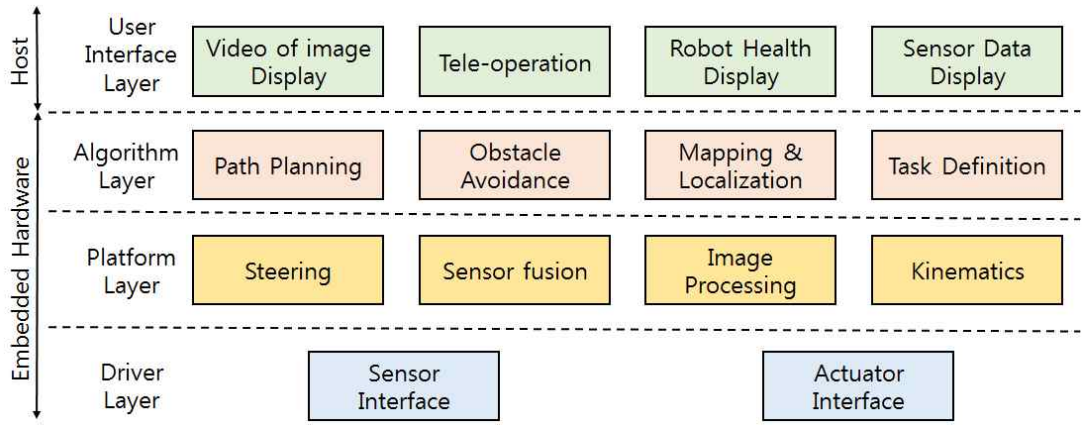
4. 사용자 인터페이스 계층의 컴포넌트에 대해 정의한다.

사용자 인터페이스 계층에는 다음과 같은 내용들이 정의되어야 한다.

첫째, 원격지에서 모바일 로봇을 특정 지점으로 이동시키고, 매니퓰레이터를 원격지에서 조종하기 위한 원격 조종 컴포넌트(tele-operation component)가 정의되어야 한다.

둘째, 원격 조종을 하기 위해서는 원격지에서 카메라 정보를 볼 수 있어야 한다. 이를 위해서 이미지 디스플레이 컴포넌트(image display component)가 정의되어야 한다.

셋째, 로봇의 상태를 파악하기 위해서 센서 데이터를 출력시키는 컴포넌트(sensor data display component)가 정의되어야 할 필요도 있다.



[그림 3-5] 모바일 로봇을 위한 로봇 소프트웨어 아키텍처의 계층 구조

4 개발할 로봇 시스템에 대한 사용 사례를 분석한 후 컴포넌트를 결정하고 컴포넌트들 간의 상호 관계를 나타내는 컴포넌트 다이어그램을 작성한다.

1. 사용 사례들을 분석한 후 로봇 소프트웨어를 몇 개의 컴포넌트로 구성할 것인지를 결정한다.

컴포넌트는 시스템을 구성하는 세부 기능을 하나의 모듈로 구현하기 위한 단위를 의미한다. 컴포넌트로 구현할 때에는 배포가 가능하도록, 즉 다른 로봇을 개발할 때에도 사용이 가능하도록 범용성 있게 구현하여야 한다.

OPRoS와 같은 로봇 소프트웨어 아키텍처에서는 배포 가능한 컴포넌트들을 일부 만들어 두었다. 따라서 로봇 소프트웨어의 컴포넌트를 정의할 때에는 OPRoS와 같은 로봇 소프트웨어 아키텍처를 참조할 필요가 있다.

<표 3-1>은 재난 구조용 모바일 로봇을 위한 원격 시동, 바퀴 제어, 장애물 감지, 맵 생성에 관한 기능에 대한 컴포넌트에 대한 설명을 나타낸다. 해당 컴포넌트들은 OPRoS 소프트웨어 아키텍처에서 예제로 제공하는 컴포넌트들을 이용할 수 있다.

<표 3-1> 재난 구조용 모바일 로봇의 컴포넌트의 예

컴포넌트 이름	설명
KeyInputComponent	원격 제어기에서 버튼을 눌러 모바일 로봇의 시동을 켠다.
UltrasonicSensorComponent	초음파 센서를 이용하여 장애물을 감지한다.
WheelControllerComponent	모바일 로봇의 바퀴를 제어한다.
GridMapComponent	장애물이 감지되면 맵을 생성한다.
ControllerComponet	모바일 로봇의 메인 제어기로서 각 컴포넌트의 기능을 통합한다.

출처: OPRoS(2012. 3. 29). OPRoS 튜토리얼-초고주파 센서를 이용한 자율 주행. http://ropros.org에서 2016. 9. 30. 검색.

2. 각 컴포넌트들의 컴포넌트 프로파일과 인터페이스 프로파일을 정의한다.

컴포넌트 프로파일에는 컴포넌트가 가져야 하는 속성을 정의하며, 인터페이스 프로파일에는 컴포넌트가 다른 컴포넌트들과 어떤 방법으로 상호 관계를 가지는지를 정의한다.

<표 3-2>은 휠 컨트롤러 컴포넌트의 컴포넌트 프로파일의 예를 나타낸다. 바퀴 제어(wheel controller) 컴포넌트의 컴포넌트 프로파일에는 구동축 바퀴 지름 크기와 같은 바퀴에 대한 사양이나 바퀴의 최대 이동 속도와 이동 가속도와 같은 wheel의 속성들이 정의되어야 한다. 다른 컴포넌트들의 컴포넌트 프로파일도 아래와 표와 같이 정의를 해야 한다. 참고로, 아래의 휠 컨트롤러 컴포넌트는 OPRoS 홈페이지에서 다운로드받을 수 있다.

<표 3-2> wheel controller 컴포넌트의 컴포넌트 프로파일의 예

컴포넌트 프로파일 이름	설명
APIName	Wheel 제어 서비스를 제공하기 위한 알고리즘이 구현되어 있는 API 이름
WheelDiameter	구동축 바퀴 지름 크기
AxleDistance	구동 축간 거리
VarianceDistance	직진 거리 오차
VarianceDirection	회전 각도 오차
SafeRaidius	외형 크기에 대한 반지름
MaximumVelocity	최대 이동 속도
Acceleration	이동 가속도

출처: OPRoS(2012. 3. 29). OPRoS 튜토리얼-Actuaopr 구조 및 원리. http://ropros.org에서 2016. 9. 30. 검색.

<표 3-3>은 휠 컨트롤러 컴포넌트의 인터페이스 프로파일의 예를 나타낸다. OPRoS에서는 인터페이스 프로파일을 서비스 포트 프로파일이라 부르기도 한다. 바퀴 제어 컴포넌트의 인터페이스 프로파일에는 로봇의 현재 위치를 설정하거나 획득할 수 있는 기능, 일정 거리를 이동시키거나 일정 각도를 회전시키는 기능, 로봇을 정지시키는 기능 등이 정의되어야 한다. 다른 컴포넌트들의 서비스 포트 프로파일도 아래와 표와 같이 정의를 해야 한다.

<표 3-3> Wheel Controller 컴포넌트의 인터페이스 프로파일의 예

인터페이스 프로파일 이름	설명
SetPosition	로봇의 현재 위치를 설정할 수 있다.
GetPosition	로봇의 현재 위치를 획득 할 수 있다.
DriveWheel	일정 속도로 로봇을 계속 이동시킨다.
MoveWheel	일정 거리를 이동시킨다.
RotateWheel	일정 각도를 회전시킨다.
StopWheel	로봇을 정지시킨다.
IsWheeling	모터의 동작 유무를 확인할 수 있다.

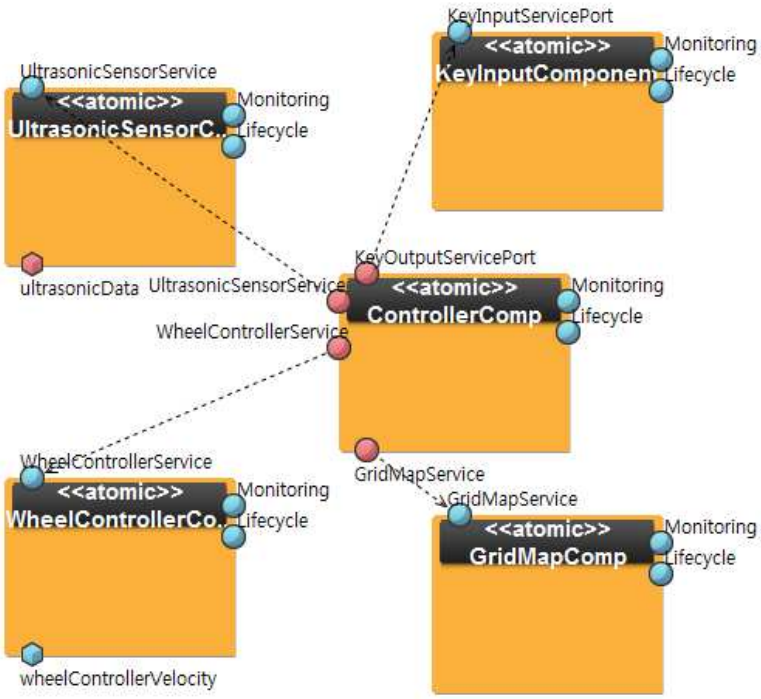
출처: OPRoS(2012. 3. 29). OPRoS 튜토리얼-Actuaopr 구조 및 원리. http://ropros.org에서 2016. 9. 30. 검색.

3. 각 컴포넌트들 간의 상호 관계를 고려하여 컴포넌트 다이어그램을 도식화한다.

[그림 3-6]은 OPRoS에서 표현한 모바일 로봇을 위한 컴포넌트 다이어그램의 예를 나타내고 있다. 점선 화살표는 각 컴포넌트들 간의 관계를 나타내고 있다. 그림에서 Ultrasonic Sensor 컴포넌트는 ultrasonicData를 data port를 이용해 보낸다. Wheel Controller 컴포넌트는 wheelControllerVelocity를 input port를 통해 받도록 하였다. 이러한 기능이 수행되도

록 Ultrasonic Sensor 컴포넌트에서는 컴포넌트 프로파일로 ultrasonicData, 인터페이스 프로파일로 GetSensorData를 정의하여야 한다. 또한 Wheel Controller 컴포넌트에서는 컴포넌트 프로파일로 wheelControllerVelocity, 인터페이스 프로파일로 MoveWheel을 정의하여야 한다.

Controller 컴포넌트는 Ultrasonic Sensor 컴포넌트로부터 초음파 센서 값(ultrasonicData)을 받아서 장애물 감지 여부와 장애물 회피 방법을 결정한 후 Wheel Controller 컴포넌트로 휠의 속도 값(wheelControllerVelocity)을 보내서 모바일 로봇을 제어하게 한다.



출처: OPRoS(2012. 3. 29). OPRoS 튜토리얼-초고주파 센서를 이용한 자율 주행. http://www.oooo.re.kr에서 2016. 9. 30. 검색.
[그림 3-6] OPRoS에서의 모바일 로봇을 위한 컴포넌트 다이어그램

수행 tip

- 로봇의 특성에 따라 로봇 소프트웨어 기능은 많이 차이가 난다. 따라서 로봇 하드웨어에 대한 면밀한 분석을 거친 후 소프트웨어 컴포넌트를 정의하여야 한다.
- 컴포넌트 다이어그램의 경우 UML 개발 도구를 이용하여 도식화할 수 있다. 이때 컴포넌트와 인터페이스의 모양이 다르므로 해당 개발 도구에 맞추어서 변경하면 된다. 본 학습모듈에서는 컴포넌트 다이어그램은 OPRoS 표현 방식으로, 나머지 다이어그램은 StarUML 표현 방식으로 도식화하였다.

학습 1	로봇 소프트웨어 요구 사항 정의하기
학습 2	로봇 소프트웨어 사용 사례 작성하기
학습 3	로봇 소프트웨어 아키텍처 정의하기

학습 4 로봇 소프트웨어 아키텍처 설계하기

4-1. 로봇 소프트웨어 아키텍처 설계

학습 목표

- 소프트웨어 컴포넌트를 이루는 클래스의 구성과 글래스 간의 상관관계를 설계할 수 있다. (class diagram)
- 소프트웨어 컴포넌트 간의 데이터 교환 및 메시지 호출 관계를 도식화하여 설계할 수 있다. (class diagram)
- 소프트웨어 컴포넌트 간 이벤트의 발생 관계를 시간 흐름에 따라 도식화 할 수 있다. (activity diagram or state diagram)
- 소프트웨어 컴포넌트 간 상호 관계 및 이벤트 흐름을 요구되는 수준에 따라 단순화된 설계도로 작성할 할 수 있다.
- 설계된 로봇 아키텍처가 내부 운영체제, 하드웨어, 미들웨어 및 외부 시스템과의 연동에서 오류가 없음을 검증할 수 있다.
- 설계된 로봇 아키텍처의 목표 성능 달성 여부를 미들웨어 및 운영체제와의 연동 메커니즘으로 검증할 수 있다.
- 설계된 로봇 아키텍처의 목표 성능 달성 여부를 사용 시나리오로 검증할 수 있다.

필요 지식 /

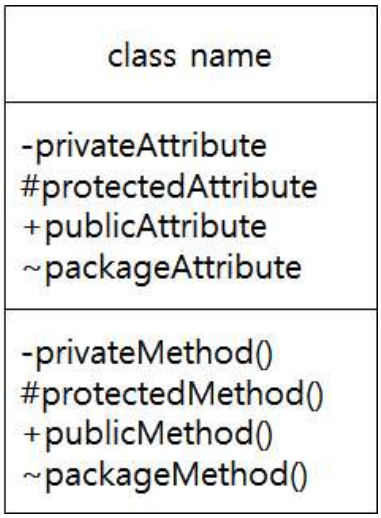
[1] 클래스 다이어그램

1. 클래스의 구성 요소

클래스의 구성 요소로는 클래스 이름(class name), 속성(attribute), 메서드(method) 등이 있다. 클래스는 [그림 4-1]과 같이 표현된다. 클래스에서는 해당 구성 요소의 외부 노출 여부를 결정할 수 있다(이를 가시성 또는 투명성(visibility)라 한다). 즉, 프로그램 내의 여러 클래스나 함수들에서 해당 구성 요소에 접근할 수 있는지 결정할 수 있다는 의미이다.

+(public) 기호는 외부에서 자유롭게 접근할 수 있다는 속성 또는 메서드라는 뜻이며,

-(private) 기호는 내부에서만 선언되어 외부에서는 이용할 수 없는 속성 또는 메서드라는 뜻이며, #(protected) 기호는 자신에게서 상속된 자식 클래스는 접근할 수 있지만 외부에서는 접근할 수 없는 속성 또는 메서드라는 뜻이며, ~(package) 기호는 같은 패키지 안에 있는 모든 클래스에서는 접근할 수 있지만 그 외의 외부에서는 접근할 수 없는 속성 또는 메서드라는 뜻이다.



[그림 4-1] 클래스의 구성 요소

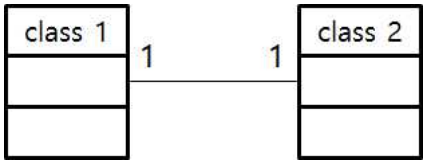
2. 관계의 표현

시스템 안의 객체는 서로 관계를 맺고 메시지를 주고받는 상호 관계를 통하여 시스템의 기능을 제공한다. 따라서 설계 작업에서는 객체가 속하는 클래스 사이의 관계를 찾아내어 표시해야 한다. 클래스 사이에는 대표적으로 연관, 집합, 상속이라는 세 가지 관계가 있다.

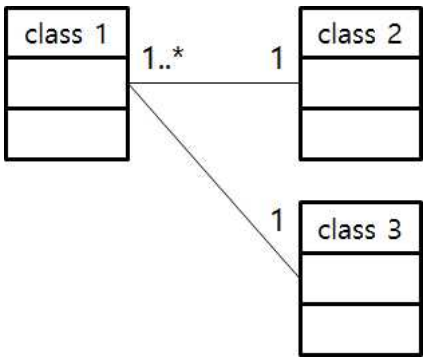
(1) 연관 관계

하나의 클래스가 다른 클래스와 관련이 있다는 뜻이다. 연관 관계는 하나의 클래스가 여러 개의 클래스와 연관 관계를 맺을 수도 있다.

[그림 4-2(a)]에서 클래스 1은 클래스 2와 연관 관계를 맺고 있음을 표현한다. 선 위의 1은 1개의 클래스와만 연관 관계를 맺고 있다는 것을 표현한다. 반면 [그림 4-2(b)]의 클래스 1은 1..*로 표현하여 1개 이상의 클래스와 연관을 맺고 있음을 표현한다. 만약 2개의 클래스와 연관 관계를 맺고 있다면, 1..2로 표현하면 된다. 클래스 2와 클래스 3은 각각 1개의 클래스와만 연관을 맺고 있기에 클래스 1과만 연관 관계에 있다고 표현한다.



(a) 클래스들 간의 연관 관계 표현



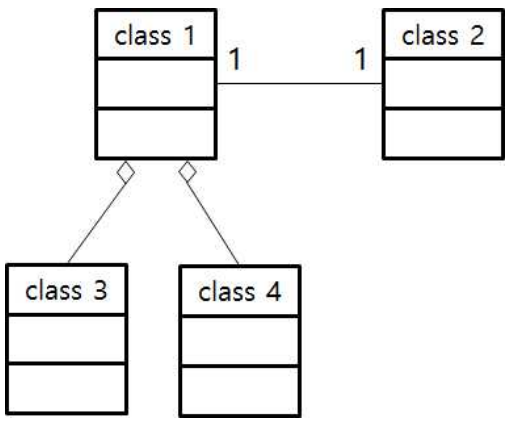
(b) 클래스들 간의 다중 관계 표현

[그림 4-2] 연관 관계에 대한 UML 표현

(2) 집합 관계

집합 관계는 두 클래스 간의 관계로서, 한 클래스가 다른 클래스의 부분이 될 때를 말한다. [그림 4-3]에서 클래스 1과 클래스 2는 연관 관계에 있다. 반면, 클래스 3과 클래스 4는 클래스의 1과 집합 관계에 있다고 표현한다.

집합의 개념은 다음과 같은 예로 생각해 볼 수 있다. 예로 클래스 1을 차량이라고 하면, 차량의 중요한 속성 중의 하나는 브레이크와 윈도우가 있다. 또 차가 운행하기 위해서는 차량의 하부 시스템인 샤시 시스템과 바디 시스템이라는 클래스가 필요하다. 샤시 시스템의 속성에는 브레이크가 들어갈 수 있으며, 바디 시스템의 속성에는 윈도우가 들어갈 수 있다. 바퀴와 핸들 클래스는 차량 클래스의 부분집합이 된다.

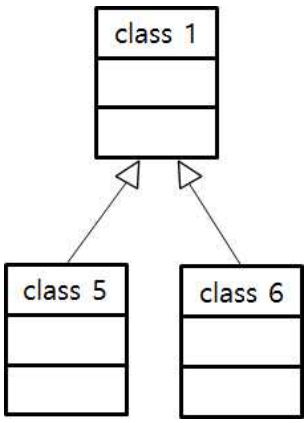


[그림 4-3] 집합 관계에 대한 UML 표현

(3) 상속 관계

상속 관계는 두 클래스 간의 관계로서, 한 클래스가 다른 클래스의 속성과 메서드를 물려받는 것을 말한다. [그림 4-4]에서 클래스 5와 클래스 6은 클래스 1과 상속 관계에 있다고 표현한다.

상속의 개념은 다음과 같은 예로 생각해 볼 수 있다. 예로, 클래스 1을 차량이라고 하면, 차량의 속성에는 브레이크와 윈도우가 있다. 클래스 5를 승용차라고 하고, 클래스 6을 트럭이라고 한다면, 클래스 5와 클래스 6은 차량이 가진 다수의 속성과 메서드를 바로 사용할 수 있다. 즉, 브레이크와 윈도우는 승용차와 트럭에서도 사용되는 속성이다. 그러나 승용차에서만 사용되는 충돌 방지 시스템 등과 같은 것은 클래스 5만 가지는 고유의 속성이 된다. 이러한 이유로 클래스 1은 모든 차량에서 사용되는 속성과 메서드들만을 기재해야 한다.



[그림 4-4] 상속 관계의 UML 표현

2 상태 다이어그램

1. 상태 다이어그램

상태 다이어그램은 이벤트 기반으로 시스템의 행위를 표현하는 다이어그램이다. 즉, 외부 이벤트에 대해 객체들이 어떻게 반응하는가를 보여주기 위한 다이어그램이다. 상태 다이어그램은 실시간 시스템(real-time system) 또는 이벤트 기반 시스템(event-driven system)을 모델링하는 데 매우 유용하다. 여기서 이벤트 기반 시스템이란 것은 모바일 로봇과 같이 어떤 이벤트가 발생할 때 어떤 일을 수행하는가를 정의한 시스템을 말한다.

소규모의 펌웨어 시스템의 경우 클래스 다이어그램을 작성하지 않고 상태 다이어그램만으로도 펌웨어 프로그램을 개발하기에 충분할 때도 있다.

2. 상태와 이벤트, 전이

(1) 상태(state)

상태(state)란 객체가 존재할 수 있는 조건 중의 하나로서, 특정 시간 간격 동안에 특정한 형태로 존재하고 있는 것을 의미한다. 예로 모바일 로봇에서 장애물을 감지한 상태와 장애물을 감지하지 않은 상태가 있을 수 있다. 충돌을 감지한 상태는 초음파 센서가 장애물을 감지한 상태인 반면, 충돌을 감지하지 않은 상태란 초음파 센서가 OFF인 상태라고 볼 수 있다.

상태에서 중요한 것은 객체가 가질 수 있는 모든 가능한 경우가 상태로 파악되어야 한다는 점이다. 즉, 객체는 파악된 상태들 이외의 상태를 가질 수 없어야 한다. 또 객체는 특정 순간에는 오직 하나의 상태로만 존재해야 한다.

상태에는 시작 상태와 종료 상태가 있다. 시작 상태는 상태 다이어그램의 시작점으로 모든 상태 다이어그램에서 존재한다. 반면, 종료 상태는 특정한 이벤트가 발생되었을 때 상태 다이어그램이 종료되는 상태로 있을 수도 있고 없을 수도 있다.

상태 다이어그램에서 상태는 [그림 4-5]와 같이 둥근 모서리를 가진 사각형이나 원형 또는 타원형으로 표시한다.

(a) 상태 (b) 시작 상태 (c) 종료 상태

[그림 4-5] 상태의 UML 표현

(2) 이벤트(event)와 전이(transition)

상태는 외부의 자극에 의하여 다른 상태로 바뀔 수 있다. 이때 객체의 상태가 다른 상태로 변경되는 것을 전이(transition)라고 부르며, 객체의 전이를 유발하는 자극을 이벤트(event)라고 부른다. 이벤트가 발생하여 상태가 전이되면 그에 대응되는 작업이 수행되어야 한다. 이것을 활동(action)이라고 한다. 활동은 이벤트에 반응하여 시스템이 수행하는 작업을 말한다. 상태 다이어그램에서 전이는 [그림 4-6]과 같이 상태와 상태 사이를 연결하는 화살표로 표시한다. 또 이벤트는 화살표 아래에 표시한다.

[그림 4-6] 이벤트와 전이의 UML 표현

3 활동 다이어그램 작성

1. 활동 다이어그램

활동 다이어그램은 펌웨어 시스템의 실행과 행위의 흐름을 표현하기 위한 다이어그램이다. 활동 다이어그램은 제어 순서도(flowchart)와 모양이 비슷하다. 다만, 제어 순서도는 문제를 해결하는 데 필요한 논리적인 흐름을 나타내지만, 활동 다이어그램은 이를 객체의 관점에서 도식화한 것이다.

활동 다이어그램은 주로 유즈 케이스 수준에서 비즈니스 프로세스를 표현하거나, 유즈 케이스 내부에 대한 구체적인 흐름을 표현하기 위하여 사용된다. 또 활동 다이어그램은 설계 단계에서 클래스 내부 메서드에 대한 알고리즘이나 구체적인 로직을 표현하기 위하여 사용될 수도 있다.

펌웨어 시스템에서 독립적인 제어 흐름을 가지는 경우 독립적인 활동 다이어그램으로 나타내어야 한다. 따라서 하나의 펌웨어 시스템에서 여러 개의 독립적인 활동 다이어그램을 가질 수 있다. 또 이러한 독립적인 활동 다이어그램 간의 상호 작용에 대해서도 표현을 할 수 있다.

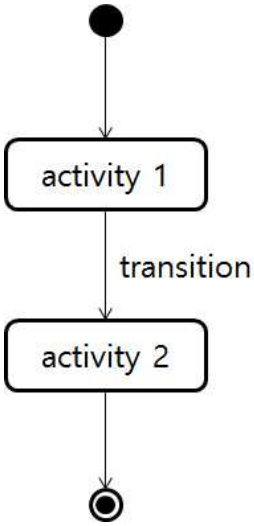
활동 다이어그램은 순차 다이어그램이나 상태 다이어그램에서는 나타내기 어려운 상황을 표현할 수 있는 다이어그램으로, 조건에 따른 활동의 선택이나 병행적인 활동 등과 같은 활동의 흐름을 나타내는 데 사용되거나 사용자 인터페이스를 위하여 사용되는 화면들 간의 흐름 관계를 나타내는 데 사용될 수 있다. 활동 다이어그램은 상태 다이어그램과 마찬가지로 소규모의 펌웨어 시스템을 설계할 때 자주 이용된다.

2. 활동(activity)과 천이(transition)

(1) 활동과 천이

활동은 펌웨어 시스템에서 특정한 일들의 처리와 실행을 의미하며, [그림 4-7]과 같이 모서리가 둥근 사각형으로 나타낸다. 활동 다이어그램은 검은색 동그라미로 표현되는 시작점이 있으며, 이중 동그라미로 표현되는 종료점이 있다. 이는 상태 다이어그램의 시작점 및 종료점과 동일하다.

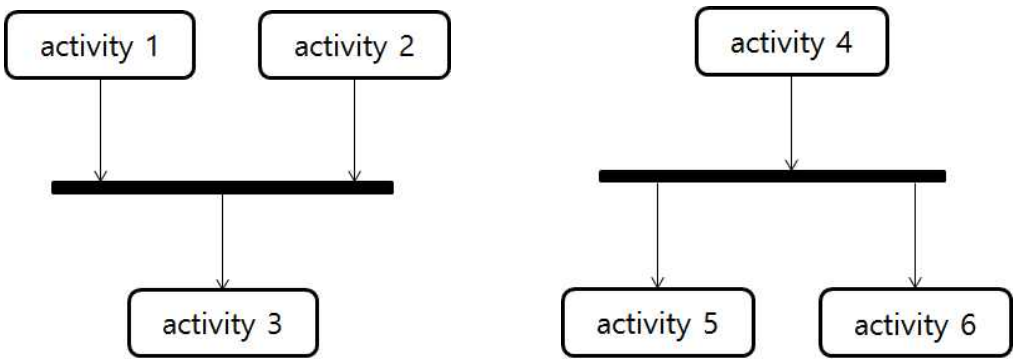
활동 다이어그램에서는 하나의 활동이 처리되고 나면 다음의 활동으로 자동적으로 옮겨지게 되는데, 이를 천이라고 한다. 천이는 상태 다이어그램과 마찬가지로 화살표로 표현한다. 일반적으로 활동 다이어그램은 활동의 선후에 따라 수직 방향으로 작성한다.



[그림 4-7] 활동과 천이의 UML 표현

(2) 결합(join)과 분기(fork), 결정(decision)과 병합(merge)

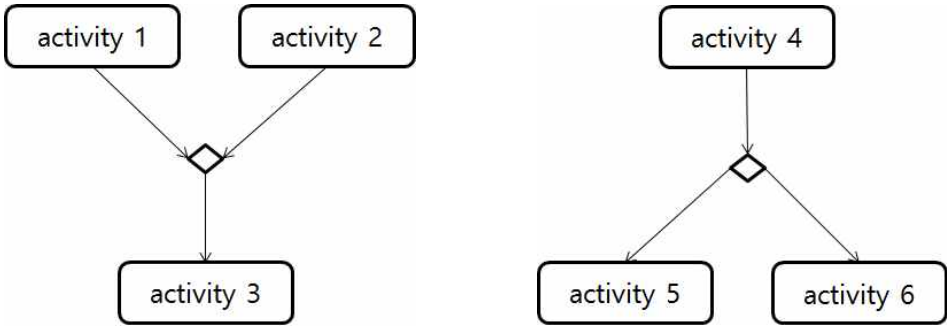
결합과 분기는 [그림 4-8]과 같이 동기화 막대(synchronization bar)를 사용하여 표현한다. 여기에서 동기화 막대는 2개 이상의 활동이 동시에 마쳐야 하거나, 동시에 시작해야 한다는 의미이다. 결합은 2개의 활동이 동시에 끝난 경우에 다음 활동으로 넘어간다는 의미이며, 분기는 2개의 다음 활동이 동시에 시작한다는 의미이다.



(a) 결합 (b) 분기

[그림 4-8] 결합과 분기의 UML 표현

결정과 병합은 [그림 4-9]와 같이 제어 구조를 뜻하는 마름모꼴을 사용하여 표현한다. 여기에서 제어 구조는 2개의 활동 중에서 하나를 선택한다는 의미이다. 결정은 여러 개의 활동이 종료되고 나면 다음 활동이 시작한다는 의미이며, 병합은 하나의 활동이 끝나고 난 후, 다음 활동 중에서 하나가 선택된다는 의미이다.

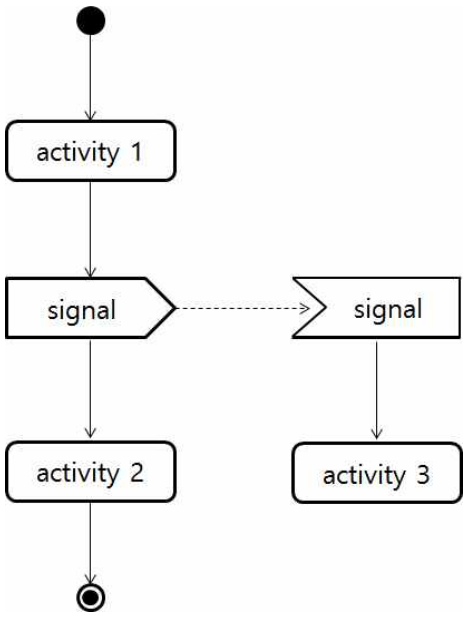


(a) 결정 (b) 병합

[그림 4-9] 결정과 병합의 UML 표현

(3) 신호(signal)

신호는 독립적인 활동 다이어그램 간의 상호 작용을 표현하기 위하여 사용된다. 신호는 [그림 4-10]과 같이 뾰족한 오각형 모양의 송신 시그널과 쐐기 모양으로 파인 다각형의 수신 시그널로 표현된다.



[그림 4-10] 신호의 UML 표현

수행 내용 / 소프트웨어 아키텍처 설계하기

재료 · 자료

- 로봇 하드웨어 사양서
- 로봇 운영체제 사양서
- 로봇 기능 정의서
- 로봇 소프트웨어 프레임워크
- 로봇 소프트웨어 아키텍처 요구 사항 정의서
- 소프트웨어 아키텍처 접근법
- 소프트웨어 아키텍처 설계 지침서
- 오픈소스 지적 재산권 지침서

기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 로봇 소프트웨어 프레임워크 개발 환경툴(SDK)
- 프로그램 개발용 소프트웨어
- 컴파일러
- 문서 작성용 소프트웨어
- 소프트웨어 품질 검사 도구

안전 · 유의 사항

- 클래스 다이어그램, 상태 다이어그램, 활동 다이어그램을 작성할 때에는 하드웨어 성능과 소프트웨어 아키텍처를 고려하여야 한다.
- 클래스 다이어그램, 상태 다이어그램, 활동 다이어그램을 도식화할 때 개발에 사용되는 UML 개발 도구에 대한 정보를 사전에 이해하여야 한다.

수행 순서

1 로봇 소프트웨어 아키텍처와 사용 사례를 분석한 후 각 컴포넌트를 위한 클래스 다이어그램을 작성한다.

1. 사용 사례와 컴포넌트 다이어그램으로부터 클래스 후보를 찾아낸다.

사용 사례와 컴포넌트 다이어그램으로부터 클래스 후보를 찾아낸다. 어떠한 것을 클래스로 할 것인가는 매우 어려운 문제이다. 특히, 클래스는 실제 존재하는 것일 수도 있고 추상적인 것일 수도 있다. 예로 모바일 로봇에서 모터나 센서는 실제 존재하는 형태이지만, 경로 계획 알고리즘과 같은 것은 추상적인 형태이다.

클래스 후보를 찾을 때는 클래스 이름을 기재하여야 한다. 클래스 이름은 클래스들 간에 중복되어서는 안 되며 직관적으로 이해하기 쉬운 이름을 사용하는 것이 좋다. 또 첫 번째 글자를 대문자로 시작하여야 한다.

예로 <표 4-1>과 같이 모바일 로봇에서 초음파 센서는 UltrasonicSensor로, 경로 계획 알고리즘은 PathPlanning으로 표현할 수 있다. 컴포넌트별로 1개의 클래스로만 표현할 수도 있으며, 2개 이상의 클래스로 표현하는 게 유리할 수도 있다. 예로, 초음파 센서 컴포넌트의 경우 UltrasonicSensor이라는 1개의 클래스만 정의하면 되며, 컨트롤러 컴포넌트의 경우 장애물에 관련된 클래스와 주행에 관련된 클래스로 정의되어야 할 것이다.

<표 4-1> 모바일 로봇 시스템을 위한 클래스 후보

컴포넌트	클래스 후보	설명
Wheel Controller	WheelMotor	바퀴를 회전시키기 위한 구동 모터
	SteerMotor	방향 전환을 위한 조향 모터
	Brake	로봇을 정지시키기 위한 브레이크
Ultrasonic Sensor	UttrasonicSensor	장애물을 감지하는 초음파 센서
Controller	Collision	장애물에 관련된 결정을 하는 알고리즘
	Driving	주행에 관련된 결정을 하는 알고리즘

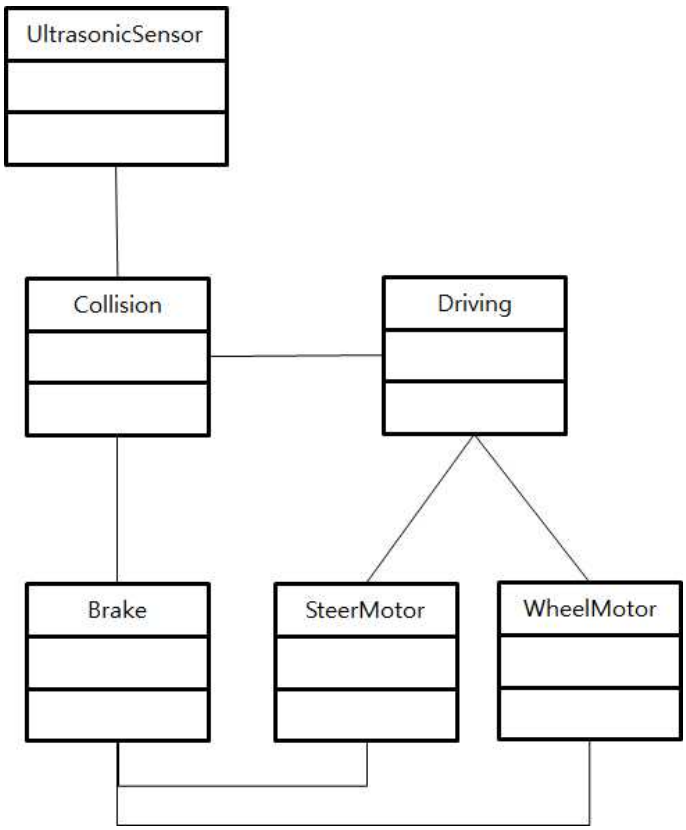
2. 위의 클래스 후보들 간의 관계를 파악한다.

클래스 후보를 찾은 이후에는 클래스들 간의 상호 관계를 파악하여야 한다. 상호 관계에는 앞에서 언급한 바와 같이 연관 관계, 집합 관계, 상속 관계가 있다. 두 클래스 후보들 간에 동사로 표현된다면 연관 관계에 있다고 할 수 있다.

이렇게 연관 관계를 파악하다 보면, 연관 관계에 얽매이지 않는 클래스 후보가 생길 수 있다. 이러한 클래스 후보들은 하나의 클래스로 간주하기 보다는 집합 관계에 있는 클래스로 만들던가, 아니면 특정 클래스의 속성이나 메서드에 포함시키는 것도 한 가지 방법이다.

예로 UltrasonicSonic 클래스와 Collision 클래스는 장애물이 감지하는 동작을 하기 때문에 연관 관계가 있다. 또 Wheel Controller 클래스는 Driving 클래스에 의하여 입력으로 받아 들여져야 하기 때문에 연관 관계가 있다.

클래스 후보들 간의 관계를 파악하고 나서, [그림 4-11]과 같은 연관 관계를 도식화한다. 이러한 클래스 다이어그램은 최적화된 형태는 아니며, 개발자들과 수요자 간의 다양한 논의를 통해 최적화시켜 나가야 한다.



[그림 4-11] 모바일 로봇을 위한 클래스 후보의 연관 관계의 예

3. 클래스 후보들의 속성과 메서드들을 파악한 후, 클래스 다이어그램을 완성한다.

상호 관계를 맺는 클래스 후보들을 선정하고 나면, 마지막 단계로 각 클래스 후보를 위한 속성과 메서드를 추가한다. 속성의 경우, 소유격에 따라 나오는 구나 형용절은 속성일 가능성이 많다. 또 시스템에 의해 저장될 필요가 있는 것들은 모두 속성으로 간주하여야 한다. 메서드는 클래스가 행하는 동사에 해당된다.

예로 WheelMotor 클래스의 경우 속성으로는 WheelMotor 등이 있을 수 있으며, 메서드로는 setPosition( ), getPostion( ), driveWheel( ) 등이 있을 수 있다. SteerMotor 클래스의 경우 속성으로는 SteerMotor 등이 있을 수 있으며, 메서드로는 rotateWheel( ) 등이 있을 수 있다. Brake의 경우 메서드로 stopWheel( ) 등이 있을 수 있다.

이러한 방법으로 <표 4-2>와 같이 클래스 후보들의 속성과 메서드들을 파악해본다.

<표 4-2> 모바일 로봇을 위한 속성과 메서드의 예

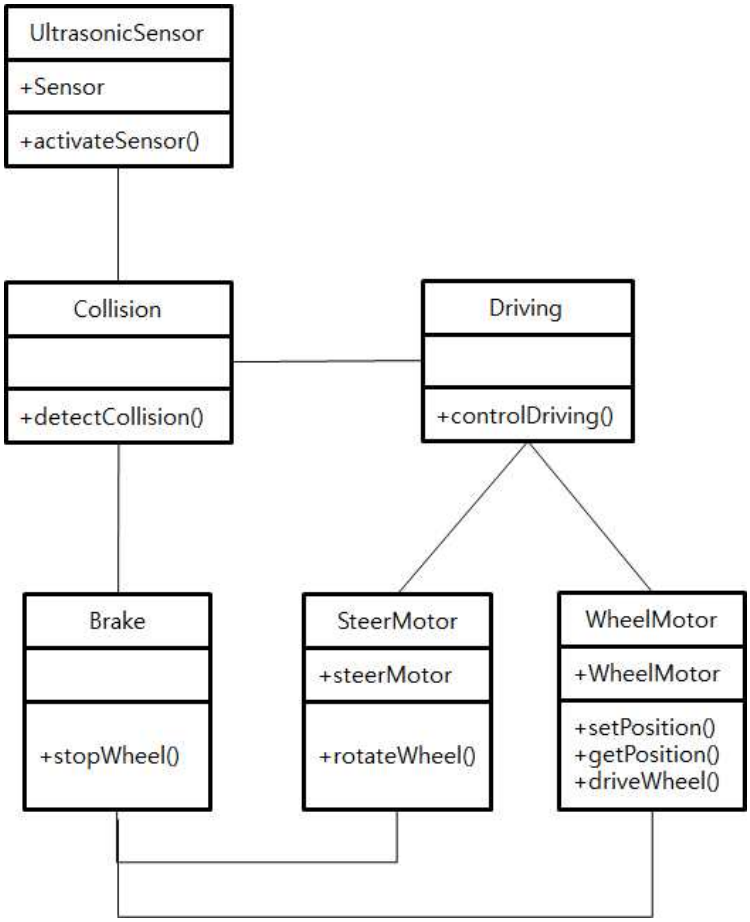
클래스	속성 또는 메서드	설명
WheelMotor	+WheelMotor	바퀴 구동 모터
	+setPosition( )	로봇의 현재 위치를 설정한다.
	+getPosition( )	로봇의 현재 위치를 획득한다.
	+driveWheel( )	일정 속도로 로봇을 이동시킨다.
SteerMotor	+steerMotor	조향 모터
	+rotateWheel( )	일정 각도를 회전시킨다.
Brake	+stopWheel( )	로봇을 정지시킨다.
UltrasonicSensor	+Sensor	초음파 센서
	+activateSensor( )	초음파 센서가 동작한다.
Collision	+detectCollision( )	충돌을 감지한다.
Driving	+controlDriving( )	주행을 제어한다.

4. UML 설계 도구를 이용하여 클래스 다이어그램을 도식화한다.

클래스 후보를 찾고 클래스 후보들 간의 상호 관계를 규명하고, 클래스 후보들의 속성과 메서드들을 추가하는 절차를 거치면서 무수히 많은 클래스 후보들의 유효성이 밝혀질 것이다.

클래스 다이어그램은 프로그램을 하기 전에 전체적인 밑그림을 그리는 작업이므로(건축에서는 스케일 모델을 만드는 작업이기에), 개발자들 간에 많은 토론을 거쳐 최적의 클래스와 속성, 메서드들을 찾아야 한다. 불필요한 클래스나 속성, 메서드들은 사용되지 않는 코드들을 남발함으로써 프로그램을 지저분하게 만든다. 반면, 찾아내지 못한 클래스나 속성, 메서드들이 존재하는 경우 프로그램에서 각 기능들을 구현하는 데 있어 여러 가지 어려움에 직면하게 될 수도 있다.

클래스, 속성, 메서드들은 클래스 다이어그램을 통하여 일차적으로 확정을 짓게 되지만, 이후 진행되는 행위 모델링 과정(즉 순차 다이어그램, 상태 다이어그램, 활동 다이어그램)을 거치면서 추가적으로 클래스가 발견되기도 하고, 확정된 클래스의 효용성이 떨어지기도 한다. 물론, 속성과 메서드들이 추가적으로 발견되기도 하고, 확정된 속성과 메서드들의 효용성이 떨어지기도 한다. 이러한 일들이 발생하면 클래스와 속성, 메서드들을 수정하여야 한다. 이러한 절차를 거쳐 속성과 메서드들을 가진 클래스들을 모아서 UML 설계 도구를 이용하여 [그림 4-12]와 같은 클래스 다이어그램을 완성한다.



[그림 4-12] 모바일 로봇을 위한 클래스 다이어그램의 예

2 클래스 다이어그램으로부터 이벤트와 상태를 정의한 후 상태 다이어그램을 작성한다.

1. 클래스 다이어그램으로부터 이벤트 후보를 선정한다.

클래스 다이어그램은 객체의 입장에서 바라보는 시각이지만, 상태 다이어그램은 이벤트의 입장에서 바라보는 시각이다. 클래스 다이어그램과 상태 다이어그램을 비교해 보자.

예로 모바일 로봇에서 ultrasonic sensor에 의하여 외부 장애물을 감지하고, wheel motor를 이용하여 로봇을 움직인다. 이러한 것을 메서드라고 한다.

반면 상태 다이어그램에서는 모바일 로봇의 상태의 관점에서 바라볼 수 있다. ultrasonic sensor의 경우 초음파 센서를 이용하여 지속적으로 장애물이 있는지를 관찰한다. 만약, 장애물이 감지되면 로봇을 제어하기 위한 여러 가지 활동들을 수행하게 된다. 여기에서 초음파 센서가 동작한다는 것은 UltrasonicSensor 클래스의 activateSensor( ) 메서드가 동작했다는 의미이며, 장애물이 감지되었다는 것은 detectCollision( ) 메서드가 동작했다는 의미이다.

이러한 방법으로 <표 4-3>과 같이 이벤트를 결정한다.

<표 4-3> 모바일 로봇을 위한 이벤트의 예

구분	이벤트명
이벤트	• activate_ultrasonic_sensor • drive_wheel • rotate_wheel • stop_wheel • detect_collision

2. 사용 사례와 컴포넌트 다이어그램을 분석하여 몇 개의 상태로 나눌 것인지를 결정한다.

상태는 이벤트에 의해서 일시적으로 발생되는 추상적인 개념이므로 결정하기가 쉽지 않다. 상태를 너무 광범위하게 선정하면 상태 내부에서 일어나는 메서드들을 상태 다이어그램으로는 표현하기가 어려우며, 상태를 너무 협소하게 선정하면 상태들 간의 전이를 표현하기가 어려울 수 있다.

따라서 상태 다이어그램에서는 관심의 대상이 되는 외부 자극, 즉 이벤트들이 무엇인지를 정확하게 파악하고, 관심의 대상이 되는 이벤트들을 중심으로 상태를 결정하는 것이 좋다. 예로 모바일 로봇에서 주 관심의 대상은 로봇의 주행과 장애물 감지이다. 따라서 <표 4-4>와 같이 로봇이 정지해 있는 상태를 waiting 상태로, 정상적으로 동작하는 상태를 driving 상태로, 장애물이 감지되고 난 이후의 상태를 collision 상태로 볼 수 있다.

<표 4-4> 모바일 로봇을 위한 상태의 예

구분	상태명
상태	• waiting • driving • collision

3. 이벤트 후보들 중에서 시스템이 안정적으로 동작하기 위해서 필요한 상태와 이벤트들을 선택한다.

이벤트 후보들 중에는 상태의 전이와 관계되지 않는 이벤트들도 있다. 이러한 이벤트 후보들은 이벤트에서 제외되어야 한다. 또 이벤트들 중에서는 시스템의 안정적 동작을 방해하는 이벤트들도 있다. 이러한 이벤트들이 발생되지 않도록 상태와 이벤트들을 설정하여야 한다.

특정 상태는 2개 이상의 이벤트가 발생될 경우에만 진입하는 상태도 있다. 이를 도식화할 수 있는 상태 다이어그램을 그려야 한다.

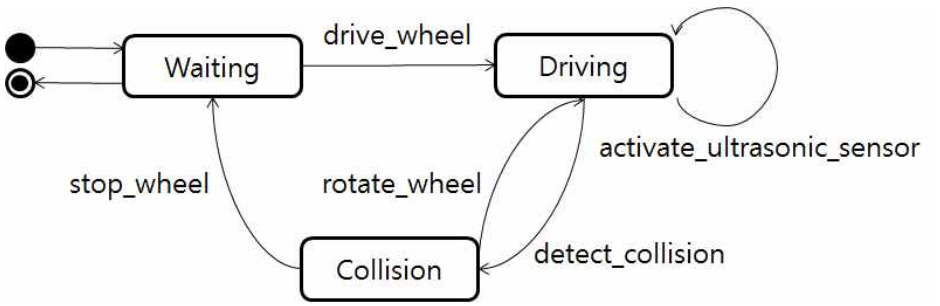
<표 4-5>는 시스템이 안정적으로 동작하기 위하여 필요한 모바일 로봇을 위한 상태와 이벤트의 예를 나타낸다.

<표 2-5> 모바일 로봇을 위한 상태와 이벤트

구분	상태 또는 이벤트명
상태	• waiting • driving • collision
이벤트	• activate_ultrasonic_sensor • drive_wheel • rotate_wheel • stop_wheel • detect_collision

4. UML 설계 도구를 이용하여 상태 다이어그램을 도식화한다.

이벤트 후보를 찾고 이벤트들 중에서 펌웨어 시스템의 주된 관심이 무엇인지를 파악한 후에 상태를 결정하고 나면, 상태와 이벤트가 확정된다. 이러한 절차를 거쳐 상태와 이벤트들을 모아서 [그림 2-5]와 같은 상태 다이어그램을 완성한다.



[그림 4-13] 모바일 로봇을 위한 상태 다이어그램의 예