한국정보과학회

제5권 제1호
Vol. 5 No. 1

KIISE · KIPS
Joint Workshop on Software Engineering Technology 2007
(KSEJW-2007)

한국정보과학회 · 한국정보처리학회
소프트웨어공학연구회
한국 소프트웨어공학기술 합동 워크샵 2007 발표집

2007년 8월 23일(목)~24일(금)

전라남도 신안군 증도 엘도라도리조트

사단법인 한국정보과학회 소프트웨어공학연구회

사단법인 한국정보처리학회 소프트웨어공학연구회

- 주 최 : 한국정보과학회, 한국정보처리학회
- 주 관 : 한국정보과학회 소프트웨어공학연구회
  한국정보처리학회 소프트웨어공학연구회
- 후 원 : KAIST 소프트웨어프로세스개선센터

목 차

주제 1 : 소프트웨어 개발방법

주제 2 : 시스템 & 개발도구

주제 3 : 분석 및 모델링

확장된 xUML을 사용한 장애물 회피용 소형 무인차 모델링 연구*

(A Study on Modeling the Obstacle Avoidance UGV using Extended Executable UML)

김 예 진† 손 현 승‡ 김 우 열§ 서 진 원** 김 동 호††
(Yaejin Kim) (Hyunseung Son) (Wooyeol Kim) (Jinwon Seo) (Dongho Kim)
서 윤 숙‡‡ 류 동 국§§ 김 영 철***
(Yunsuk Seo) (Dongkuk Ryu) (R. Youngchul Kim)

요 약 현존하는 SW 모델링 언어들로서는 임베디드 SW의 효율적인 모델링을 위한 요소가 부족하다. 본 논문에서는 기존의 UML x.x버전들이 임베디드 소프트웨어를 모델링 하는데 적합한지를 비교/분석한 결과, 모델 자체가 코드처럼 수행 가능한 통합 모델링언어인 xUML(Executable UML)[2,4,7]을 채택하고 임베디드 소프트웨어 모델링에 기존의 xUML의 부족한 면을 보완 및 확장하였다. 확장된 xUML 노테이션은 병렬과 실시간 처리, 그리고 비결정적 상태 다이어그램도 표현이 가능하도록 제안하였다. 사례 연구로서 장애물 회피용 소형 무인차(Small Unmanned Ground Vehicle)가 동작하도록 실시간 임베디드 시스템의 모델링과 수행을 보여주었다.

1. 서 론

국방 및 산업용 로봇은 위험하고 인간이 접근하기 어려운 환경에서의 여러 가지 작업을 위하여 운동성 및 신뢰성이 뛰어난 무인 로봇의 개발이 요구되고 있다. 특히 국방용 로봇은 일반적인 산업용 로봇과는 달리 동적 안정성과 다기능적인 작업영역을 가진 로봇이 필요하다. 그리고 근래에는 로봇 핵심이 하드웨어 기술에서 소프트웨어 기술로 이동하고 있다. 초창기 로봇 소프트웨어는 간단한 제어 프로그램만으로 산업용 로봇을 제어하는데 그쳤으나, 최근에는 멀티미디어 처리나 항공 시스템등과 같은 복잡하고 커다란 시스템까지도 제어하고 있다. 이렇듯 복잡한 기능을 제공하는 소프트웨어를 좀 더 효율적이며 안정적으로 개발하기 위해 소프트웨어를 모델링 하는 연구가 활발히 진행 중이다[1].

보통 일반적인 소프트웨어를 모델링하기 위해서 UML 을 사용한다. 기존의 UML x.x 은 소프트웨어 시스템을 모델링하기 위한 언어로서 사용하며 소프트웨어 개발자들은 시스템에 대한 모델을 만들 수 있다[2]. 하지만 복잡하고 주위환경에 신속하게 대처해야 하는 실시간 임베디드 시스템은 서비스의 질(QoS or Quality of Service), Low-level 프로그래밍 그리고 안전성과 신뢰성에 대한 특별한 고려가 필요하다[3].

일반적인 소프트웨어의 모델링과는 달리 모델 자체가 코드처럼 수행 가능한 통합 모델링언어를 xUML(Executable UML)이라 한다[2,4,11]. xUML 은 기존의 UML x.x 에 실행과 관련된 개념(executable semantics)들과 시간에 관련된 규칙(timing rules)들을

* 이 논문은 2006학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음
† 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
yaejin@selab.hongik.ac.kr
‡ 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
sonhs @selab.hongik.ac.kr
§ 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
john@selab.hongik.ac.kr
** 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
jinwon@selab.hongik.ac.kr
†† 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
kdh@selab.hongik.ac.kr
‡‡ 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
jyun@selab.hongik.ac.kr
§§ 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
ryu@selab.hongik.ac.kr
*** 홍익대학교 컴퓨터정보통신 소프트웨어공학연구실
bob@selab.hongik.ac.kr

더한 것이다. xUML 의 모델은 실행과 테스트, 디버깅이 가능하며, 시스템 성능 측정까지 가능하다.

본 논문에서는 기존의 UML x.x 버전들과 xUML 이 실시간 임베디드 소프트웨어를 모델링 하는데 적합한지를 비교/분석한 후, 임베디드 소프트웨어 모델링에 xUML 을 적용하고자 부족한 면을 보완 및 확장하고자 한다.

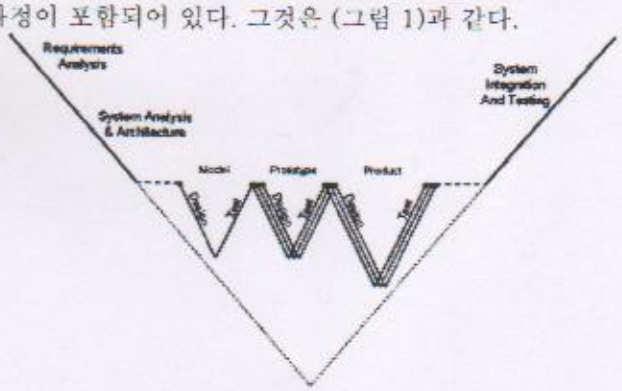
본 논문의 구성은 다음과 같다. 제 2 장에서는 관련연구로서 Multiple V-model 과 Executable UML 에 대해 알아보고 xUML 과 각 버전별 UML 을 비교/분석한다. 제 3 장에서는 제안한 xUML 의 확장에 대해 설명한다. 제 4 장에서는 확장된 xUML 이용하여 모델링 과정과 절차를 설명한다. 5 장에서는 적용사례로 장애물 회피용 소형 무인차를 모델링 한다. 마지막으로 제 6 장에서는 결론 및 향후연구를 언급한다.

2. 관련연구

2.1. Multiple V-model

Multiple V-Model 은 잘 알려진 V-model 을 기반으로 한다. 원칙적으로 각 산출형태(모델, 프로토타입, 최종 산출물)는 디자인과, 빌드 그리고 테스트 활동을 포함하는 완전한 V 개발 주기를 따른다.

Multiple V-모델은 임베디드 시스템 개발에 적절한 모형으로 기존의 V 모델을 모델, 프로토타입, 최종 제품에 관한 세 가지로 나누어 연결한 모델이다. Multiple V-모델의 처음 단계인 모델 단계는 PC 를 통해 요구된 시스템의 행위를 모의 실험하는 단계이다. 그리고 이러한 모델 단계가 적합하다고 판정되면 프로토타입 단계에서는 모델로부터 소스코드를 생성하고 실험용 하드웨어에 소스코드를 삽입하여 점차적으로 실제 하드웨어로 전환해 가면서 최종 제품으로 개발해 간다. 그리고 각 단계는 디자인, 구현, 테스트의 순차적인 개발 과정이 포함되어 있다. 그것은 (그림 1)과 같다.



(그림 1) Multiple V-Model 개발 주기

Multiple V-모델은 임베디드 시스템 개발하기에 적합하다. 왜냐하면 앞서 설명한 바와 같이 Multiple V-모델은 임베디드 시스템의 개발 단계인 모델, 프로토타입, 최종 제품과 동일하게 세단계로 구분하여 각 단계의 특성에 따라 테스트 디자인 기법이나 테스트 단계 등을 적용할 수 있기 때문이다. 또한 요구 사항에 변화나 오류로 인한 변경이 생길 경우 최종 산출물 보다는 프로토타입 단계, 그리고 프로토타입 단계보다는 모델 단계에서 변경함으로써 시간과 비용을 모두 감소 시키는 효과도 기대할 수 있다.

2.2. xUML(Executable UML)

UML 은 객체 관련 표준화기구인 OMG 에서 1997 년 11 월 객체모델링기술(OMT)[6], OOSE 방법론[7] 등을 연합하여 만든 통합 모델링 언어로 객체 지향적 분석·설계 방법론의 표준 지정을 목표로 하고 있다. UML 사용자 가이드는 UML 에 대해 " UML 이란 소프트웨어 시스템의 산출물을 가시화하고, 명세하고, 구축하고, 문서화하기 위한 그래픽 언어이다" 라고 정의하였다[9].

Executable UML[7,8]은 기호로 스펙을 설명하는 언어이다. 기존의 UML(Unified Modeling Language) 1.x 표기법에 " 실행에 관련된 개념(executable semantics)들" 과 " 시간에 관련된 규칙(timing rules)들" 을 더한 것이다. 기존의 모호한 모델과 달리, Executable 모델은 시간에 관련된 문제와 동기화에 관련된 문제, 그리고 자원의 획득과 이용에 관련된 문제들을 자세히 서술한다. 결론적으로, Executable UML 은 복잡한 실시간(realtime) 분산(distributed) 임베디드(embedded) 시스템의 스펙을 서술하기에 적합하며 많이 이용된다.

<표 1>은 xUML 의 장점을 파악하기 위해 각기 다른 버전의 UML 들을 비교해 놓은 것이다. UML 은 크게 분석 능력과 설계 능력, 구현 능력, 그리고 테스트/디버깅 능력으로 나누어 비교할 수 있다. 우선 xUML 은 분석 능력에서 이벤트를 모델링하고 구현 전에 설계 변화를 분석할 수 있다는 장점이 있다. 설계 능력으로는 동적 모델링에서 중요시 되는 동시발생 문제를 표현하고 모델과 구현의 일관성이 유지된다는 장점이 있다. 또한 설계 변경이 용이하므로 여러 플랫폼(타겟)에서 재사용이 가능하고, 설계 도중 문제를 발견 및 수정할 수 있다. 구현 능력으로는 xUML 은 기존의 스켈레톤 코드가 아닌 동적 요소를 포함한 완벽한 코드가 발생된다는 것이다. 마지막으로 테스트/디버깅 능력은

실행 모델이기 때문에 디자인 레벨에서의 디버깅이 가능하다.

<표 1> UML 버전별 비교[2]

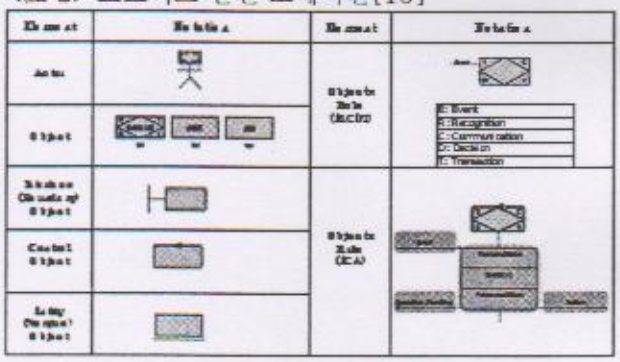
능력 구분	UML 1.x	UML 2.0	Embedded UML	xUML
주요 시장	상업적/ 비즈니스 어플리케이션	상업적/ 비즈니스 어플리케이션/ 실시간 모델	SoC	임베디드/ 실시간 모델
분석 능력	×	○	○	○
이벤트 모델링	×	○	-	○
구현 전 설계 변화 분석	×	○	-	○
설계 능력	○	○		○
Reverse Fork/Join	×	×	×	×
선택적 Concurrency	×	×	×	×
설계와 구현 사이의 일관성이 유지	×	○	×	○
설계 변경이 용이	×	×	-	○
여러 타겟에서의 설계 재사용	×	○	-	○
설계 단계에서 문제를 발견 및 수정	×	○	-	○
구현 능력	×	×	×	○
모델과 코드의 연관	×	○	×	○
실시간/임베디드 시스템의 구현능력	×	○	-	○
테스트/디버그 능력	×	○	×	○
실행 모델	×	○	×	○
디자인 레벨에서의 디버깅	×	○	×	○

하지만 이러한 xUML 조차도 Reverse Fork/Join 이나 선택적 병행성과 같은 실시간 임베디드 소프트웨어를 모델링 하는데 한계점이 존재한다. 그래서 다음 장에서 확장된 xUML 을 제안하였다.

3. 확장된 xUML

2 장에서 언급한 xUML 조차도 표현하지 못하는 부분이 있기에 본 논문에서는 xUML 을 확장하였다. <표 2>과 같이 확장된 xUML 에서 클래스 다이어그램은 역할(Role)과 규칙(Rule)을 포함하며 시스템의 정적 구조를 묘사한다. 객체는 ERCDT(Event/Recognition/Communication/Decision/Transaction)의 구조를 가지고 있다. 이는 객체가 각자 다른 역할을 수행하는 것을 의미한다. 인터페이스 객체는 이벤트(Event)가 들어오면 인지(Regognition)해서 통신(Communication)하는 ERC 의 구조를 가진다. 그리고 제어 객체는 인터페이스의 기능뿐만 아니라 결정(Decision)을 내리고 수행(Transaction)하는 ERCDT 의 모든 구조를 가질 수 있다. 또한 서비스 객체는 이벤트(Event)가 들어오면 수행(Transaction)을 하게 되는 ET 의 구조를 가지게 된다. 이벤트가 들어오면 객체 내부에서 조건에 맞는지 검사를 한 후, 조건에 맞으면 액션을 수행하게 되고 맞지 않으면 예외처리를 하게 된다.

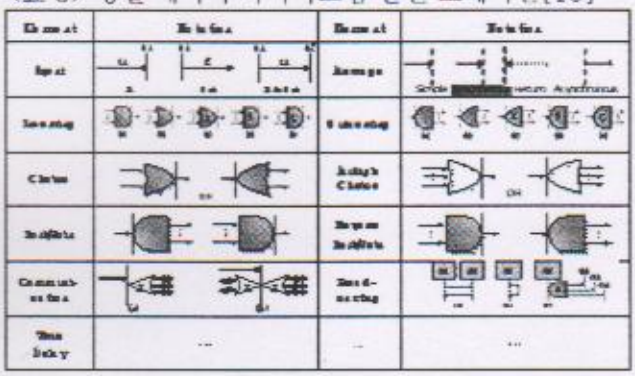
<표 2> 오브젝트 관련 노테이션[10]



병렬 메시지 다이어그램(CMD: Concurrent Message Diagram)은 기존의 시퀀스 다이어그램에 실세계에서 발생될 수 있는 병렬 메커니즘(fork-join, reverse fork-join 등)을 포함하도록 확장한 것이다. <표 3>은 확장된 xUML 의 노테이션을 묘사한 것이다. 우선 확장된 xUML 은 동기 메시지를 기본으로 한다. 그리고 클래스 다이어그램에서 정의한 것과 마찬가지로 각각의 객체에 역할(Role)이 있다. 이는 어떠한 이벤트가

들어오면 객체가 인지를 하고 통신을 하여 제어객체가 결정을 내리면 수행하게 되는 것이다. 이 모든 역할을 객체 내에 표현해 줌으로써 각 객체들의 역할을 한눈에 파악할 수 있다. 또 다른 객체의 특징으로는 객체 내부의 규칙을 가지고 있다는 것이다. ECA(Event/Condition/Action) 법칙을 따르기 때문에 이벤트가 들어오면 객체 내부에서 조건에 맞는지 검사를 한 후, 조건에 맞으면 액션을 수행하게 되고 맞지 않으면 예외처리를 하게 된다.

<표 3> 병렬 메시지 다이어그램 관련 노테이션[10]



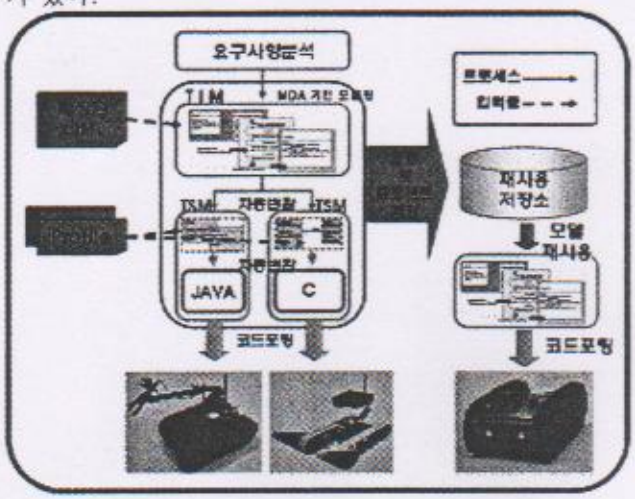
병렬 상태 다이어그램(CSD: Concurrent State Diagram)은 기본적으로 OCL(Object Constraint Language)을 포함하여 서술된다. 그리고 이 다이어그램은 Deterministic/Stochastic 메커니즘을 포함한다. 향후 Non-deterministic 상태를 Deterministic 상태로 자동 변환할 수 있도록 연구를 진행 중이다.

4. 확장된 xUML을 사용한 모델링

4.1. 개발 과정

MDA 기반의 임베디드 소프트웨어 개발 프로세스[5,10]를 도식화 하면 (그림 2)와 같다. 이는 요구사항 및 분석 과정을 거쳐 확장된 xUML을 사용하여 타겟 독립 모델을 설계한다. 타겟 독립 모델에 운영체제 및 프로세서를 고려한 프로파일을 적용하여 각자 타겟 시스템에 알맞은 모델인 타겟 종속 모델로 변환된다. 타겟 종속 모델은 타겟 시스템에 맞는 코드로 자동 변환 시켜 준다.

이후 생성된 소스코드를 타겟 모델에 탑재하는 작업이 바로 개발한 소프트웨어를 임베디드 소프트웨어 시스템에 적용시키는 것이다. 이때 생성된 소스코드와 모델은 재사용 저장소에 저장되고 후에 기능을 변경하거나 추가하여 새로운 임베디드 시스템을 구축하고자 한다면 저장된 모델을 재사용하여 개발 시간을 단축시킬 수 있다. 최종적으로는 모델 단계에서의 재사용을 통해 우리의 최종 목표인 상호운용성 또한 높일 수가 있다.



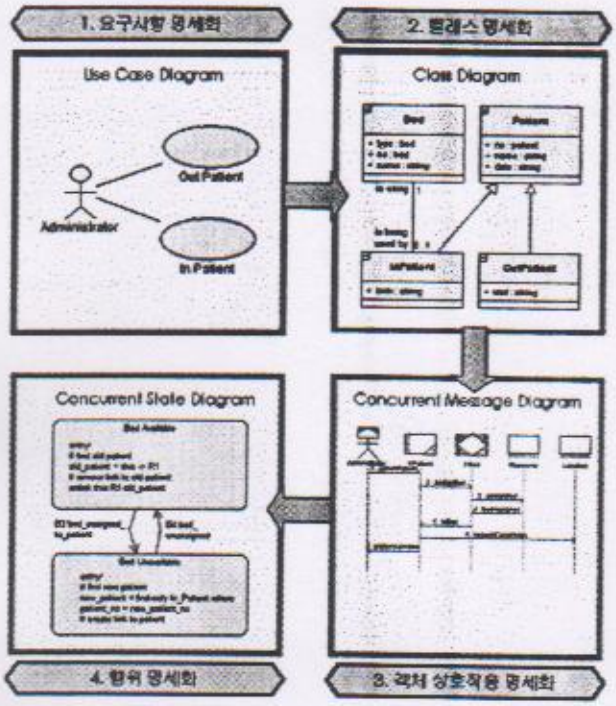
(그림 2) MDA 기반 개발 프로세스

4.2. 모델링 절차

소프트웨어 모델의 주된 목적은 정보 표현이라고 할 수 있겠다. 소프트웨어 모델은 구축할 소프트웨어를 추상적인 수준에서 표현하여 줌으로써, 보는 이로 하여금 복잡한 소프트웨어를 이해 할 수 있도록 도와준다[13].

모델링 언어를 모바일 임베디드 소프트웨어의 표현수단으로도 사용하고 시스템을 설계하는데도 사용한다. 또한 사용하는 모델링언어인 확장된 xUML은 구현 레벨의 코드까지 매핑이 되어있으므로 모델 구축만으로 시스템 구축이 가능하다[12].

도구를 이용하여 모델링 언어로 표현하기 위해서는 (그림 3)처럼 총 4 단계를 가진다.



(그림 3) 확장된 xUML 을 이용한 모델링 순서

첫 번째 단계인 " 요구사항 명세화" 는 Use Case Diagram 을 이용하여 표현한다. 이 단계에서 는 요구사항에 대한 분석과 모바일 기기가 가져야 될 특성과 사용자들의 요구사항이 무엇인지를 분석한다.

두 번째 단계인 " 클래스 명세화" 는 Class Diagram 을 이용하여 표현한다. 첫 번째 단계로부터 분석된 자료를 가지고 객체와 클래스를 추출하여 시스템의 구조를 설계한다.

세 번째 단계인 " 객체 상호작용 명세화" 는 Concurrent Message Diagram 을 이용하여 표현한다. 두 번째 단계에서 설계한 Class Diagram 을 참고하여 객체들 간의 상호작용을 모델링 하고 역할을 정의한다.

마지막 단계인 " 행위 명세화" 는 Concurrent State Diagram 을 이용하여 표현한다. 객체의 동적인 움직임을 모델링한다. 이 단계의 작업이 시스템을 구동시키는 역할을 해준다.

5. 적용사례

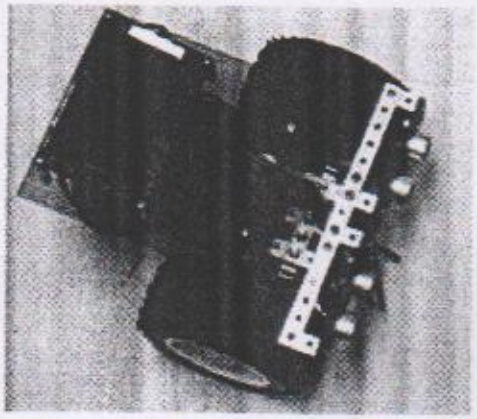
5.1. 적용 대상

(1) 장애물 회피용 소형 무인차

장애물 회피용 소형 무인차는 목표 지점까지 장애물을 회피하면서 전진하는 로봇이다. 이 로봇은 고정된 장애물, 움직이는 장애물, 예상치 못한 장애물을 피하는 것으로 구분할 수 있다. 고정된 장애물은 물체와 시스템간의 거리를 구하여 일정거리를 기점으로 상황을 판단하는 방법을 사용한다. 움직이는 장애물은 시스템과의 물체와의 위치 섹터를 항상 유지시키도록 하여 움직이는 물체를 피하게 된다. 예상치 못한 장애물은 장애물과 센서의 값을 계속 모니터링 하여 갑작스러운 변동에 대한 대처를 하게 된다.

(2) 타겟 시스템

(그림 4)는 장애물을 회피 Vehicle 의 모습이다. 이 시스템의 하드웨어 사양은 <표 4>에 정리하였다. 이 시스템의 특징을 살펴보면 자바가상머신이 하드웨어로 내장되어 자바언어를 수행할 수 있도록 되어있다. 또한 최상위에 Text LCD 가 장착되어 시스템 내부의 동작을 LCD 를 통해 볼 수 있다. 그리고 초음파센서가 장착되어 물체에 닿지 않아도 물체와의 거리를 측정할 수 있다.



(그림 4) 장애물 회피용 소형 무인차

<표 4> Javline 하드웨어 정보

Microcontroller	Ubicom SX48AC 20MHz
RAM	32 KByte
EEPROM	32 KByte
센서	초음파센서 2 개
디스플레이	Text LCD
서보모터	2 개
JVM	하드웨어
개발 언어	Java

3) 기능적 요구사항

장애물 회피용 소형 무인차가 수행해야 하는 기능들에 대해 열거한다. 장애물 회피용 소형 무인차는 기본적으로

장애물을 탐지하고 이를 피해 우회하는 경로를 찾는 기능을 수행한다.

①이동

장애물 회피용 소형 무인차의 기본적인 이동방향은 '전진' 이다. 장애물 회피용 소형 무인차에 전원이 들어오고 프로그램이 작동되기 시작하면 가장 먼저 앞으로 이동을 시작한다. 전진은 센서가 장애물을 탐지하기 전 까지 계속 수행된다.

장애물이 탐지되면 장비는 전진을 멈춘다. 그리고 회피 알고리즘에 따라 전/후/좌/우 등의 다른 방향으로 이동이 가능해야 한다. 좌/우 와 같이 방향을 틀어야 하는 경우, 대략적인 각도의 조절이 가능해야 한다.

②장애물 탐지

장애물 회피용 소형 무인차는 전방을 탐지할 수 있는 초음파 센서가 부착되어있다. 초음파 센서를 통해 탐색장비와 전방 장애물과의 거리를 계산해낼 수 있다. 탐색 장비는 이러한 기능을 이용하여 50cm 이내에서 장애물이 탐지될 경우, 회피 상태로 전환한다.

장애물이 탐지될 수 있는 조건은 두 가지가 있다.

고정된 장애물

원래부터 그 자리에 있었으며, 움직임이 없는 장애물이다. 이 경우 탐색장비가 장애물에 접근하면서 거리를 예상할 수 있기 때문에 장애물에 충돌하기 전에 미리 회피를 할 수 있다.

예기치 못한 장애물

장애물이 이동을 하거나 갑자기 생겨난 경우로, 이러한 상황에서는 탐색 장비가 장애물과 충돌할 수 있다. 충돌한 경우에는 우회할 수 있는 거리를 확보하기 위해 후진을 할 필요가 있다.

③회피

회피는 장애물 회피용 소형 무인차와 장애물과의 거리에 따라서 두 가지로 나눠질 수 있지만, 기본적으로는 장애물이 탐지되었을 때 오른 쪽으로 우회한다는 점에서 동일하다.

장애물과의 거리가 50cm 미만인 경우 회피 상태가 되며, 장애물과의 거리에 따라서 다음 세 가지 방식으로 회피를 한다.

100cm 이상

물체가 앞에 없으므로 고속으로 전진한다.

50cm ~ 100cm

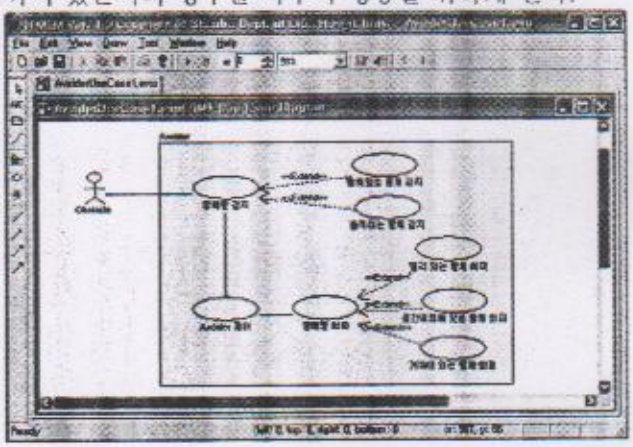
탐지된 장애물과 충돌하기 전에 회피할 수 있는 충분한 거리이다. 그러므로 충돌하지 않고 부드럽게 우회할 수 있어야 한다.

0cm ~ 50cm

장애물과 탐색장비와의 거리가 너무 가까워 이미 충돌했거나 혹은 충돌할 가능성이 있는 거리이다. 그러므로 우회할 수 있는 충분한 거리를 확보하기 위해 후진 후 우회를 해야 한다.

5.2. Use Case Diagram

적용대상의 기능적인 요구사항을 적용하여 모델링하면 (그림 5)와 같다. 장애물 로봇이 해야 되는 경우를 두 가지 케이스로 나누었는데 그것은 장애물 감지와 회피 이다. 첫 번째 유스 케이스는 <표 5>와 같고 두 번째 유스 케이스는 <표 6>과 같다. 장애물 회피용 소형 무인차가 물체를 발견하게 되면 멈춰있는 물체 인지 움직이는 물체 인지를 판별하여 장애물을 회피하게 된다. 장애물 회피 할 때는 물체가 멀리 있는지 중간에 있는지 가까이 있는지의 경우를 나누어 행동을 취하게 된다.



(그림 5) Use Case Diagram

장애물 감지의 유스 케이스를 자세히 살펴보면 물체를 감지하기 시작하면 초음파 센서로부터 거리를 측정하고 이 정보를 이용하여 로봇의 행위를 결정짓도록 한다.

<표 5> Use Case 1

Use Case	장애물 감지
Purpose	주변 상황을 인지/파악
Primary Scenario	1. 전진하면서 센서의 상태를 감시한다. 2. 50cm 이내에 장애물이 감지된다. 3. 초음파 센서로부터 거리를

	얻어온다. 4. 거리를 판단하고 우회 상태로 전이한다.
정확성	장애물과 탐색장비 사이의 거리를 정확히 측정한다.
안정성	장애물과 충돌하지 않도록 반응이 정확해야 한다.

장애물 회피 유스 케이스는 상애물 감지를 통해 얻어진 정보를 실제로 수행하는 행위 부분이다. 발견된 장애물의 위치에 따라 회피하고 앞으로 나가게 된다.

<표 6>Use Case 2

Use Case	장애물 회피
Purpose	물체를 발견하고 안전하게 물체를 회피 한다
Primary Scenario	1. 후진을 명령한다. 2. 후진한다. 3. 우측으로 30 도 회전을 명령한다. 4. 좌/우측 Wheel 의 속도를 제어한다. 5. 전진 상태로 전환한다. 6. 전진한다.
정확성	정해진 각도만큼 정확히 회전한다.
안정성	장애물을 완전히 회피할 때 까지 우회를 반복한다.

5.3. 클래스 다이어그램

시스템의 정적구조는 장착된 장비를 기준으로 나누도록 하였다. 그래서 4 개의 부분으로 나누었고 이것을 모델링 한 것이 (그림 6)이다. 4 가지 영역은 Controller, Wheel, UltraSonic, LCD 로 나누었다. 각 영역을 설명하면 다음과 같다.

1)Controller

시스템의 전체를 관할하고 각 자원에 대한 감시 및 제어를 하는 부분이다. 주로 수행하는 행위는 초음파 센서를 통해 수집된 정보들을 판별하고 각 정보에 맞는 행위를 수행 하도록 모터를 제어하고 결과를 LCD 에 표시한다.

2)Wheel

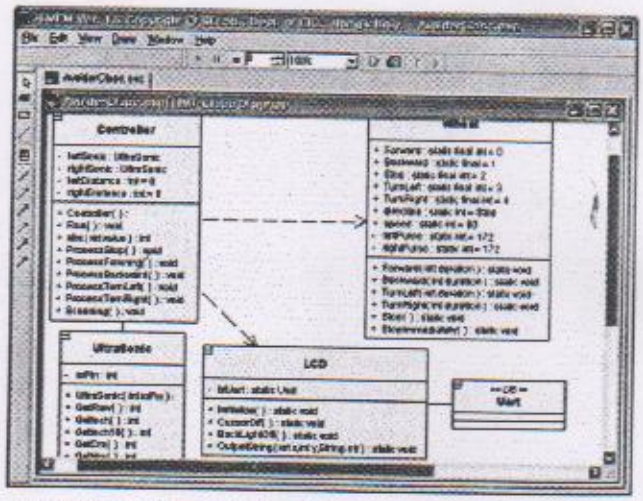
시스템의 이동 수단이 되는 바퀴에 관한 제어를 담당한다. 장애물 회피용 소형 무인차는 2 개의 서보모터를 이용하여 전진, 후진, 좌, 우를 이동할 수 있고 이들을 관리 한다. Wheel 클래스는 단순하게 제어부로부터 받은 명령만 수행하는 역할을 한다.

3)UltraSonic

시스템의 센서부로 초음파 센서의 정보를 제어부에 전달하는 역할을 한다. 초음파 센서로부터 받은 정보를 분석하여 제어부가 받아들일 수 있는 데이터로 변환하여 전달해주게 된다. 센서로부터 읽은 값은 하드웨어 수치로 일반적인 단위의 값이 아니기 때문에 이것을 cm 로 변환시켜 제어부에 넘겨주는 역할을 한다.

4)LCD

LCD 는 시스템을 상태를 표현 할 수 있는 출력 장치이다. 시스템의 센서의 값이 올바른지 검사하기 위해서 사용하기도 하고 시스템이 현재 어떤 상태인지 체크하기 위해서도 사용된다. 장애물 회피용 소형 무인차에서는 현재 시스템이 어떤 상태(전/후/좌/우)인지를 표시하는 용도로 사용 한다.



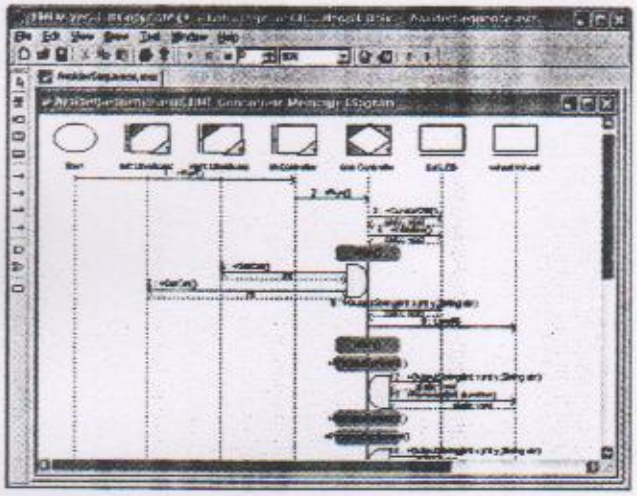
(그림 6) 클래스 다이어그램

5.4. 병렬 메시지 다이어그램

클래스 다이어그램을 통해서 시스템의 구조를 결정하면 병렬 메시지 순차 다이어그램을 사용하여 각 객체들 사이의 관계를 기술한다. 초음파 센서가 2 개이기 때문에 UltraSonic 클래스는 2 개의 객체를 생성시킨다. 그리고 Controller, LCD, Wheel 은 각각 한 개의 객체로 생성 한다. 장애물 회피용 소형 무인차의 모터는 2 개이지만 Wheel 클래스가 모터 두 개를 함께 제어하기 때문에 한 개의 객체로 생성하였다.

시스템의 작동순서는 장애물 회피용 소형 무인차에 전원이 들어와 시스템이 구동되면 LCD 화면에 초기화 되고 있다는 메시지를 보내고 그 즉시 초음파 센서들의

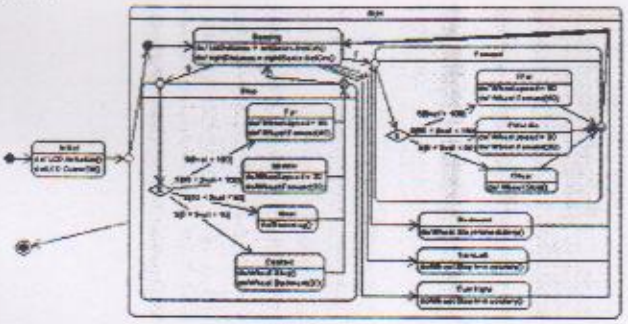
값을 불러와 시스템을 구동하게 된다. 그리고 시스템이 행위를 수행 할 때마다 LCD 화면에 상태 정보를 표시한다. (그림 7)은 장애물 회피용 소형 무인차의 병렬 메시지 순차 다이어그램으로 표현한 것이다.



(그림 7) 병렬 메시지 다이어그램

5.5. 병렬 상태 다이어그램

이번 단계에서는 객체의 상태를 기술하게 된다. (그림 8)은 Controller 객체의 상태를 모델링한 모습이다. 시스템의 모든 행위들의 제어는 Controller 에 집중되어 있기 때문에 상태 다이어그램이 매우 복잡해 질 수 있다. 이러한 복잡성을 피하기위해서 계층적인 구조로 모델링 하였다.



(그림 8) 병렬 상태 다이어그램

Controller 의 상태는 크게 초기 상태와 실행 상태로 나누었다. 그리고 이 실행 상태를 다시 Sensing, Forward, Backward, TurnLeft, TurnRight, Stop 상태로 나누었다. 이것을 계층적으로 표현하면 <표 7>과 같다.

<표 7> Controller 객체 상태의 계층적인 구조

구조			설명
Initial			초기화하는 상태
Run	Sensing		데이터 감시하는 상태
	Forward	FFar	100cm 이상
		FMiddle	50~100cm
		FNear	0~50cm
	Stop	Far	100cm 이상
		Middle	50~100cm
		Near	10~50cm
		Contact	0~10cm
	Backward		후진하는 상태
	TureLeft		왼쪽으로 도는 상태
	TurnRight		오른쪽으로 도는 상태

5.6. 코드생성

클래스 다이어그램, 병렬 메시지 순차 다이어그램, 병렬 상태 다이어그램을 통해서 코드를 생성하게 된다. 이때 생성된 코드는 (그림 9)와 같다. 여기에 자동화도구로 해결할 수 없는 코드를 작성하고 컴파일 한다. 그리고 장애물 회피용 소형 무인차에 포팅 하여 로봇을 작동시키게 된다.

(그림 9) 도구를 통해 생성된 코드

5.7. 테스트

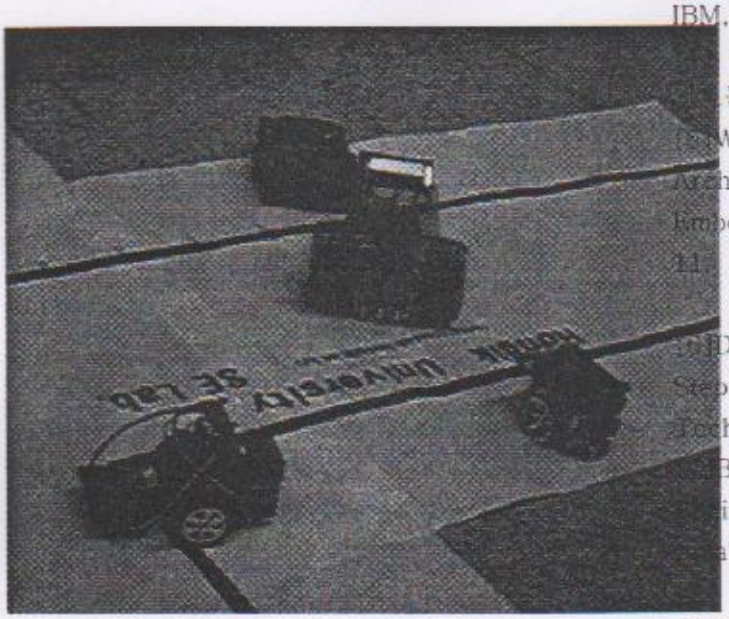
1) 테스트 케이스

도구를 통해 생성된 코드를 실제 장비에 포팅 하였을 때 요구사항에 맞게 작동되는지를 테스트하기 위해 테스트 케이스를 다음과 같이 정의 하였다.

물체의 위치가 100cm 이상 일 때
고속으로 전진한다.
물체의 위치가 50cm ~ 100cm 일 때
속도를 줄이고 우회한다.
물체의 위치가 0cm ~ 50cm 일 때
일단 정지 후 뒤로 후진하여 거리를 확보한 후 우회한다.

2) 결과 확인

장애물 회피용 소형 무인차는 앞에서 정의한 테스트 케이스에 맞도록 모두 작동되었다. (그림 10)은 장애물 회피용 소형 무인차가 장애물에 위치에 따라 작동되는 것을 찍은 모습이다.



(그림 10) 장애물 회피용 소형 무인차의 작동

5. 결 론

현존하는 SW 모델링 언어들로서는 임베디드 SW 의 효율적인 모델링을 위한 요소가 부족하다. 본 논문에서는 기존의 UML x.x 버전들이 임베디드 소프트웨어를 모델링 하는데 적합한지를 비교/분석한 결과, 모델 자체가 코드처럼 수행 가능한 통합 모델링언어인 xUML 을 채택하고 임베디드 소프트웨어 모델링에 기존의 xUML 의 부족한 면을 보완 및 확장하였다. 이는 기존의 시퀀스 다이어그램에 AND/OR/XOR 등의 개념과 Concurrency 와 Fork/Join 등의 개념을 추가함으로써 장애물 회피용 소형 무인차의 병렬과 실시간 처리, 그리고 비결정적 상태 다이어그램도 모델링이 가능하였다.

향후 연구과제로 우리가 확장한 xUML 을 간단한 로봇이 아닌 커다란 통신 시스템에도 적용되도록 연구를 진행 중이다.

참고문헌

[1] Axel Jantsch, Modeling Embedded System and SOCs, Mogan Kaufmann, 2004.

[2] 김우열, MDA 기반의 임베디드 소프트웨어 모델링에 관한 연구, 홍익대학교 석사학위논문, 2005.

[3] 이호수, 유비쿼터스의 핵심기술: 임베디드 시스템, IBM, 신기술 신경영, Vol.9, Spring, 2004.

김인기, UML 설계와 응용: 클래스 모델 만들기, 문화사, 2003.

W. Kim, R. Y. Kim, "Adapting Model Driven Architecture for Modeling Heterogeneous Embedded S/W Components," ICHIT, Vol. 2, 2006.

Derr, K.W. "Apply OMT: A Pracitcal Step-by-step Guide to Using the Object Modeling Technique", SIGS Books, 1995

Bernd Bruegge, "Object-Oriented Software Engineering: Using UML, Patterns, and ",Prentice Hall, 2004

최병삼, "모바일 컨버전스에 대한 오해와 전략적 시사점", 주간기술동향 통권 1258 호, 2006

[9] http://www.omg.org/,2003

[10]김우열, 김영철, "확장된 xUML 을 사용한 MDA 기반 이종 임베디드 소프트웨어 컴포넌트 모델링에 관한 연구", 정보처리논문지, 제 14-D 권 제 1 호, 2007. 2.

[11] Mellor, K. Scott, A. Uhl, D. Weise, " MDA Distilled" , Addison-Wesley, 2004

[12] 김우열,김영철,"실시간 임베디드 소프트웨어 모델링을 위한 xUML 확장에 관한 연구", 2006, KIPS. Vol.13,No. 1,P.231-234

[13]김영기,"UML 활용 사례연구", 2005, 한국경영과학회, P. 294-299