UML을 적용한 OHT 제어 시스템 설계

심 갑 식† · 정 태 영††

요 약

반도체 업계에서는 200mm 반도체 웨이퍼 생산공정이 300mm 웨이퍼 생산공정으로 바뀜에 따라, 300mm 반도체 웨이퍼를 이동시키는 로봇을 모니터링하고 시뮬레이션하기 위한 소프트웨어 개발이 필요하다. 이런 소프트웨어는 독립적으로 동작하는 것이 아니라 MCS와 하위 시스템인 로봇과 통신하면서 실행되어야 하므로, 그 구성이 복잡하다. 그러므로, 이 시스템을 체계적으로 개발하기 위해서는 객체지향 개발 방법론인 UML을 적용할 필요성이 있다. 본 논문은 반도체 웨이퍼를 생산공정에서 이동시키는 로봇을 모니터링하고 시뮬레이션하기 위한 소프트웨어 개발에 UML 프로세스를 적용하였다. UML을 적용하여 UML을 기반으로 한 개발 프로세스 정의와 개발 단계별 업무들에 대한 구체적인 산출물들을 만들어 내었다.

Overhead Hoist Transport Control System Design Using UML

Gab-Sig Sim† · Tae-Young Jung††

ABSTRACT

As the semiconductor industrials change 200mm-sized semiconductor wafer production process to 300mm-sized one, it requires to develop the software for monitoring and simulating the robot which transfers a 300mm-sized semiconductor wafer. Because such a software don't run at standalone but communicate MCS(Material Control System) and its subsystem a robot, its architecture is very complex. Therefore, in order to develop such a software systematically, we must utilize an object-oriented development methodology, UML. This paper presents an UML process application developing the software for monitoring and simulating the robot which transfers a semiconductor wafer on the production process.

키워드 : OHT 제어 시스템(Overhead Hoist Transport Control System), UML, Use Case, Actor

1. 서 론

정보 시스템들은 복잡화, 대형화, 전략정보시스템화 그리고 통합화로 발전하고 있다. 소프트웨어 개발에서도 이러한 현실을 반영하기 위해서는 사용자의 요구사항들을 보다 세밀하게 분석하고, 설계를 통한 전체적인 개발 공정관리와 체계성을 필요로 하게 되었다. 이것이 바로 소프트웨어 개발 방법론의 등장 배경이라 할 수 있다. 이러한 변화를 효율적으로 지원할 수 있는 소프트웨어 개발 방법론이 절실히 요구되고 있는 실정이다.

소프트웨어 프로세스는 소프트웨어와 관련된 프로젝트 계획, 설계문서, 코드, 시험사례, 사용자매뉴얼 등을 개발하고 유지보수하기 위하여 사용하는 활동, 방법, 기법 등이 일련의 순서를 가지는 집합이다. 소프트웨어 개발 및 관리를 위한 표준, 절차, 방법, 도구 및 환경을 많이 개발되고 증가 일로에 있다. 이러한 증가에 따라 특히 제품과 서비스가 함께 합쳐지는 경우 소프트웨어 엔지니어가 소프트웨어를 개발하고 관리함에 있어 "의사소통의 공통공유 수단"으로 사용할 수 있는 프레임웍을 갖출 필요성이 커지게 되었다.

이러한 필요성에 따라 OMG(Object Management Group)에서 표준 객체지향 모델링 언어로 UML(Unified Modeling Language)을 채택하였다. 객체지향 모델링의 표준 방안인 UML은 분석, 설계를 위한 표기법으로서 역할을 담당하고 있으며, UML 개발 과정은 예비 반복이라는 개괄적인 모델을 구축하는 단계로 시작하여 상세화 단계에서 점진, 반복적 개발과정을 거치며, 또 다른 관점에서는 시작, 상세화, 구축, 진화라는 4단계의 세분화 과정을 거친다. 이 UML [1-3]이 산업계 표준으로 채택되었기 때문에 여러 회사들은 이 방법론을 적용하여 소프트웨어를 개발하고 있다.

본 논문에서는 UML을 적용하여 반도체 공정에서 웨이퍼를 이동시키는 로봇을 제어하는 모니터링 시스템을 설계 및 구현한다. 2장에서는 천정 반송제어 시스템의 정의와 기

† 종신회원 : 진주산업대학교 교양학부 교수
†† 정 회 원 : 에이스코리아 대표
논문접수 : 2003년 9월 2일, 심사완료 : 2004년 1월 5일

능적인 요구사항들을 기술하며, 3장에서는 UML 개발 프로세스를 따라 설계한 산출물들을 기술한다. 그리고 마지막 4장에서는 결론 및 추후 연구방향을 제시한다.

2. OHT 제어 시스템 정의

OHT 제어 시스템은 반도체 웨이퍼 가공 공정에서 사용하는 시스템이다. 이 시스템은 반도체 웨이퍼 제작공정 중에서 웨이퍼를 이동시키는 작업을 말한다. 웨이퍼를 이동시키는 작업은 작업로봇이 천정에 매달려서 웨이퍼를 실어 나르는 작업을 수행한다. 이러한 작업을 수행하는 로봇을 사용자가 모니터링하고 시뮬레이션 하기 위한 시스템이 OHT 제어 시스템이다.

2.1 OHT 제어 시스템의 기능적 요구사항

이 시스템은 다음과 같은 몇 가지의 기능을 수행해야 한다. 첫째는 OHT 제어 시스템은 작업 명령을 내리는 MCS (Material Control System)와의 통신에서 IBSEMI 프로토콜을 준수하여 작업명령을 수행해야 한다. 둘째는 작업을 제어하는 데몬이 작업의 우선순위와 FIFO 기반의 작업제어를 수행해야 한다. 셋째는 로봇과의 무선 통신을 통하여 작업명령을 전송하거나 로봇의 현재 진행상태를 받아서 이를 갱신해야 한다. 넷째는 화면상으로 로봇의 상태를 모니터링 할 수 있어야 하며 작업의 상태정보를 로그에 기록해야 한다. 다섯 번째는 모니터링을 위한 맵을 생성하여야 한다.

2.2 OHT 제어 시스템의 데이터 요구 사항

OHT제어 시스템에서 보관되고 갱신·유지되는 데이터가 존재한다. 이들 데이터를 보관하기 위한 메모리구조가 필요하다. 이러한 데이터의 종류는 여러 가지가 존재할 수 있다. 첫째는 MCS로부터 전송되는 작업 정보를 저장하고 이를 관리하기 위한 메모리이다. 두 번째는 로봇의 현재 상태를 저장하는 메모리로서 로봇의 ID, 로봇의 상태 값, 작업 모드 등의 자료가 있다. 세 번째는 로봇의 레일(rail)을 따라서 작업을 수행하는 과정을 화면에 표시해주기 위한 데이터를 저장하는 메모리이다. 이 메모리에 저장되는 자료는 웨이퍼를 싣고 내리는 장비, 로봇의 위치, 로봇이 이동 중 멈출 수 있는 위치(stop station) 등이 있다.

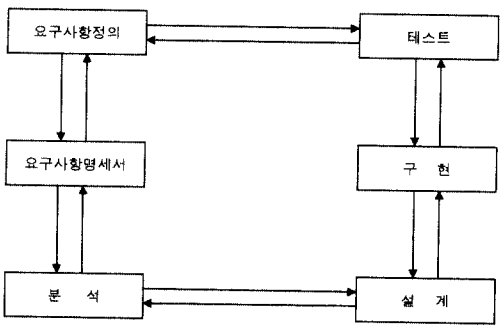
3. 개발 프로세스 및 단계별 업무 정의

본 장에서는 개발 방법론의 개발 프로세스에 따른 정의와 각 단계별 업무에 대한 정의 및 각각의 업무에 대한 산출물들을 제시한다.

3.1 어플리케이션 시스템 개발 방법론

반도체 공정 로봇 모니터링 시스템 구축을 계획하는 UML기반의 추진 방법론은 요구사항 정의, 요구사항 명세, 분석, 설계, 구현과 테스트로 구성되어 있다. 요구사항 분석부터 테스트에 이르는 개발절차 및 활동으로 분리되어 있고, 각각은 상호 밀접한 관계를 맺고 추진된다. 이러한 절차에 따르는 개발에 사용된 도구는 Rational Rose를 활용하고, 각 문서 형식 및 산출물은 MS-Office와 파워포인터를 사용한다. 또한 구현에 사용되는 언어는 Visual C++를 사용하였다. Visual Source Safe를 사용하여 버전관리를 하였다.

(그림 1)에서와 같이 반도체 공정 로봇제어 모니터링 시스템 개발영역은 여러 개발 주기를 거쳐 시스템이 개발되는데 이 개발 프로세스의 특징은 어느 한 시점에서 현재 진행 중인 프로젝트가 분석, 설계, 구현을 동시에 수행하고 있다는 것이다. 하나의 개발 주기 진행 중에 미비한 점이 있다면 새로운 개발 주기를 출발시켜 시스템의 기능성을 정의하고 이를 설계하고 필요한 구현을 완성해서 지속적인 시스템의 버전을 만들어 나가는 방법이다.



(그림 1) 개발 프로세스

3.2 요구 사항 기술

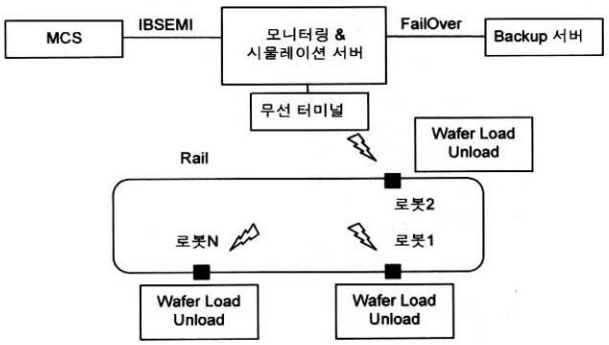
개발 절차 영역은 분석부터 개발/테스트에 이르는 활동을 요구사항 정의, 요구사항 명세, 분석, 설계, 구현, 테스트로 분류하여 패키지 분석, To-Be 시스템 제안, GAP분석, 업무 개선 및 기술구조 설계, 시스템 설계, 수정 및 보완 설계와 이에 따른 단위/종합테스트 활동을 수행하게 된다. 이러한 개발 절차 영역에서 요구사항 정의는 업무 시스템 개발에서부터 테스트 과정까지 영향을 미치게 된다. 따라서 프로젝트의 성공을 위해서는 체계적이고 명확한 요구분석이 이루어져야 하며, UML에서는 유스 케이스 모델링을 통해 사용자 혹은 고객의 요구사항을 파악하고 업무 시스템의 범위를 결정하는 방법을 제공하고 있다. 요구사항 추출은 고객이 이해할 수 있는 용어로 쓰여진 요구사항 정의문서를 작성 할 수 있도록 하는데, 요구사항 정의서(requirements definition)는 제안된 시스템에 대해서 고객이 요구하는 모든 기능에 대한 리스트이다. 이것이 고객이 원하는 것이 무엇인지 알 수 있도록 고객과 개발자간의 의사소통을 원활하게 도와주며 보통 고객과 개발자가 함께 공동으로 작성한다. 반면에 요구사항 기술서는 시스템 개발을 위해 적절

한 기술 용어를 이용하여 요구사항 정의서를 재작성 한 것이다[4]. 구축할 업무시스템의 영역을 문서로써 기술한 것을 요구사항 기술서라 하며, 작성된 요구사항 기술서는 유스 케이스 모델링 과정에서 가장 중요한 기초 자료가 된다[5]. 따라서 요구사항 기술서에는 구축되어야 할 업무 시스템의 영역이 명확히 정의되어야 하며, 그 내용 또한 사용자의 검증을 거친 것이어야 한다.

3.2.1 반도체 공정 로봇제어 시스템구조

요구사항 기술서에는 구축되어야 할 업무 시스템의 영역을 (그림 2)에서 나타내고 있다. (그림 2)는 사용자와 개발자가 구축되어야 할 시스템의 영역을 정할 수 있으며, 또한 요구사항 정의서를 기술하는데 많은 도움을 준다.

(그림 2)는 본 논문에서 구현하고자 하는 OHT 제어 시스템 구조도이다. MCS로부터 공정명령을 반도체 장비간의 표준 프로토콜인 IBSEMI를 통하여 받는다. 공정명령을 받은 시뮬레이션 서버는 현재의 로봇의 작업정보를 조회하여 가장 적합한 로봇에게 공정명령을 무선 터미널을 통하여 전송한다. 무선 터미널을 통하여 공정명령을 전송 한 후 시뮬레이션 서버의 화면 인터페이스를 통하여 실제 라인에서 움직이는 로봇을 모니터링 할 수 있다. 또한 오프라인 상태에서는 이 과정을 시뮬레이션 할 수 있는 기능을 가지고 있다.

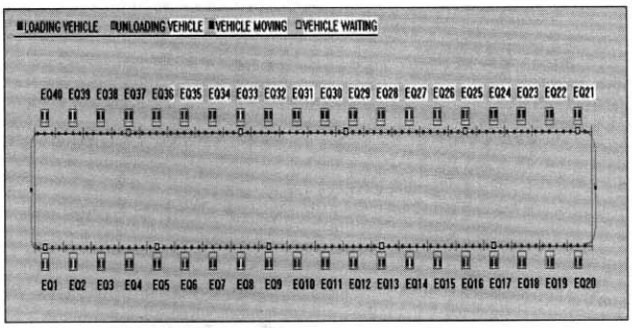


(그림 2) 시스템 구조

3.2.2 로봇 이동 경로 맵

(그림 3)은 로봇이 웨이퍼 공정에서 이동하는 라인의 모습이다. 라인에서 작은 사각형은 로봇을 나타내고, 라인상의 원은 로봇이 정지할 수 있는 위치를 나타낸다. 그리고 라인 바깥쪽의 사각형 모양은 로봇이 웨이퍼를 로딩하거나 언로딩 할 수 있는 장비위치를 의미한다.

현재의 시뮬레이션 맵에서 로봇의 개수는 4개이다. 이 4대의 로봇이 상위 서버 시스템으로부터 작업 명령을 받아서 웨이퍼를 다음 공정으로 이동시키는 작업을 수행하게 된다. 이때 작업 명령을 수행하는 로봇은 서로 충돌하지 않아야 하며, 또한 최적의 경로를 따라서 작업을 수행할 수 있어야 한다. 이러한 요구사항은 요구사항 분석을 통하여 기술되어 진다.



(그림 3) 로봇의 이동 경로를 나타내는 맵

3.2.3 요구사항 기술서

(그림 2)와 (그림 3)을 바탕으로 요구사항 기술서를 개요, 업무영역, To Be 모델 순서로 작성하면 다음과 같다.

① 개 요

OHTSC의 서브시스템인 시뮬레이션과 모니터링에 관한 기술이다. OHTME에 의해서 생성된 맵 환경설정 파일을 읽어서 공유메모리에 데이터를 쓰고, 주어진 가상의 작업 명령을 받아 작업을 수행하는 시뮬레이션 및 모니터링에 관한 요구사항 명세서이다.

② 업무영역

- 생성된 공유메모리를 초기화하기 위해서 맵 환경설정 파일을 읽는다.
- 가상의 작업 명령을 수행할 Vehicle(Robot) Thread를 생성한다.
- Vehicle(Robot) Thread는 작업 명령을 읽어서 Job Control Thread의 제어에 따라서 움직인다.
- Main Thread는 레일 경로를 따라 움직이는 Vehicle (Robot)을 그린다.
- Main Thread는 Vehicle(Robot)의 작업 상태를 화면에 그린다.
- Vehicle(Robot)의 현재 상태를 모니터링 한다.

③ To Be 모델

- OHTSM 프로세서는 시뮬레이션 및 모니터링을 하기 위한 공유메모리를 생성한다.
- OHTSM 프로세서는 생성된 공유메모리에 시뮬레이션 및 모니터링을 하기 위한 초기 데이터를 맵 파일에서 읽어 저장한다.
  - 맵 파일을 읽기 위한 DIALOG BOX를 OPEN한다.
  - 시뮬레이션용 맵 파일[*.OCF]을 선택한다.
  - DOCUMENT 클래스에서 파일을 읽기 모드로 오픈한다.
  - 시뮬레이션 관련 정보는 맵 파일을 읽으면서 공유메모리 저장구조에 맞게 파싱하여 저장한다.
  - 공유메모리에 저장된 OBJ INFO를 참조하여 맵을

화면에 보여준다.

- SIMULATION 시작하기 버튼을 누르면 시뮬레이션이 시작된다.
  - Job Control Process가 해당 작업을 Named Piped 구조에 넣어 준다.
  - 작업 명령이 입력되면 작업을 수행할 Vehicle(Robot) Thread를 생성한다.
  - Vehicle(Robot) Thread는 Job Control Thread에 의한 작업 스케줄링에 따라서 기동한다.
  - 이동 시그널을 받은 Vehicle(Robot) Thread는 Job Command에 해당하는 작업구간에 대한 표시를 화면에 디스플레이 한다.
  - Vehicle(Robot)의 위치 값을 1씩 증가시킨다.
  - 변화는 위치 값을 Vehicle(Robot) Info 공유메모리에 저장한다.
  - 위치 값을 저장 후 이벤트를 발생시킨다.
  - 이벤트가 발생하면 Main Thread는 메시지 큐에서 해당 메시지를 읽어서 Vehicle(Robot)을 그린다.
  - Vehicle(Robot)을 그릴 때 Vehicle(Robot)의 상태 값을 Vehicle(Robot) Info 공유메모리에서 읽어서 상태를 표시한다.

3.3 유스 케이스 모델

이 모델은 새롭게 혹은 기존 시스템을 기반으로 개발되거나 새로운 시스템에 대한 사용자의 요구사항을 파악하는 것이다[6]. 따라서, 도메인 전문가의 도움으로 요구사항 기술서가 작성되면 시스템을 개발하는 동안 시스템이 제공해야할 기능 또는 서비스가 유스 케이스 모델로 작성된다. 유스 케이스 모델링의 목적을 간단히 살펴보면 다음과 같다. 첫째, 시스템의 기능적 요구사항을 결정하고 설명하여 고객과 개발자간의 합의를 도출한다. 둘째, 시스템이 무엇을 할 것인가에 대한 명백하고 지속적인 설명을 통해 개발 과정에서 모든 개발자들이 요구사항에 대해 의사소통하고, 요구된 기능성을 제공해주는 상세한 설계를 위한 기반 제공한다. 셋째, 시스템을 검증하기 위한 테스트를 수행하는 기초 역할을 담당한다. 넷째, 시스템 내에서 기능상의 요구사항이 실제 클래스와 동작을 하는지에 대한 추적 능력을 제공함으로써 유스 케이스 모델을 수정함으로써 시스템을 간단하게 바꾸고 확장시킬 수 있으며, 그 후에 유스 케이스는 시스템 설계와 구현에 영향을 미친다.

유스 케이스 모델을 만드는데 실제 필요한 작업은 시스템의 정의, 액터와 유스 케이스 찾기, 유스 케이스 그리기, 유스 케이스 간의 관계 정의, 그리고 최종적으로 모델을 검증하는 과정이다. 이는 고객과 액터로 표현되는 사람들간의 토론을 포함하여 매우 상호 교류적인 형태로서, 유스 케이스 모델은 액터와 유스 케이스, 그리고 이들간의 관계를 나타내는 유스 케이스 다이어그램으로 구성된다. 본 절에서는 액터, 유스 케이스 및 관계를 추출하기 위해 요구사항 기술서로부터 유스 케이스 다이어그램의 요소를 추출한다.

3.3.1 액 터

액터는 사용자의 종류로서 시스템의 외부에 존재하면서 시스템과 직접 상호 작용하는 사람 혹은 사물이다[7]. 액터는 보통 요구사항 기술서에 '명사형'으로 표현되어진다. 액터를 고려할 때 중요한 것은 사람이나 직위보다는 '역할'을 먼저 생각해야 한다는 것이다[8]. 여기서 '역할'은 사용자와 시스템 사이의 관계를 구성하게 되는데 특유의 필요, 흥미, 기대, 행위, 책임 같은 것의 집합에 의해 정의된다[9]. <표 1>은 면담이나 요구사항 기술서 등의 방법을 이용하여 사용자의 요구사항을 파악하여 액터를 추출해 낸 하나의 예이다.

〈표 1〉 액터 목록

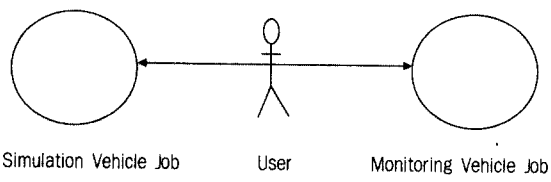
일련번호	액 터	역 할 기 술	적용규칙
1	user	OHTSM을 실행하여 시스템을 모니터링하고 시뮬레이션 하는 Operator을 의미한다	

3.3.2 유스 케이스

액터는 업무 수행을 위해 어떤 목적을 가지고 시스템을 사용한다. 이러한 액터의 목적을 위해 제공해 주어야 하는 시스템의 기능이 유스 케이스이다. 따라서 유스 케이스는 특정 액터에 대하여 관측 가능한 결과를 산출하는 일련의 이벤트 흐름으로 언제나 액터에게 어떤 가치를 제공해야 한다. 요구사항 기술서에서 액터와 시스템간의 상호작용을 찾아 서술어를 중심으로 하여 유스 케이스를 추출한다. <표 2>는 요구사항 기술서로부터 추출한 유스케이스 목록이며, (그림 4)는 유스 케이스 다이어그램이다. Simulation Robot Job 유스 케이스는 로봇의 작업을 시뮬레이션하며 Monitoring Robot Job은 모니터링 작업을 한다.

〈표 2〉 유스 케이스 목록

일련번호	유스 케이스	역 할 기 술	적용규칙
1	Simulation Robot Job	Robot의 작업을 시뮬레이션 한다.	
2	Monitoring Vehicle Job	Robot의 작업을 모니터링 한다.	



(그림 4) 유스 케이스 다이어그램

주어진 업무영역 및 시스템 영역을 유스 케이스와 액터, 그리고 그들 간의 관계로서 모두 파악하고 나면, 이제 각각의 유스 케이스에 대한 분석을 하게 된다. 각각의 유스 케이스는 여러 가지 이벤트들의 흐름으로 이루어져 있으며, 그 각각의 이벤트 흐름을 파악하는 것이 매우 중요하다. 이때 하나의 유스 케이스에 대해 가장 기초가 되는 이벤트 흐름을 기본 이벤트 흐름이라 하고 그 밖의 이벤트 흐름을 선택 이벤트 흐름이라 한다. 이러한 이벤트 흐름을 문서로 작성한 것이 유스 케이스 명세서라고 한다. <표 3>은 Simulation Vehicle(Robot) Job의 유스 케이스 명세서를 기술한 것이다.

〈표 3〉 유스 케이스 명세서

Title : Use Case Specification-[Simulation Vehicle (Robot) Job]

[1] Brief Description

이 Use Case는 Operator가 Simulation 메뉴를 선택함으로써 시작된다.

모니터링에 필요한 공유메모리는 미리 생성되어진 공유메모리 객체를 참조한다.

Rail Info Shared Memory :
2-1의 기본사항 Rail Info Shared Memory 생성 flow가 수행된다.

Vehicle Info Shared Memory :
2-2의 기본사항 Vehicle Info Shared Memory 생성 flow가 수행된다.

Object Info Shared Memory :
2-3의 기본사항 Object Info Shared Memory 생성 flow가 수행된다.

Job Info Shared Memory :
2-4의 기본사항 Job Info Shared Memory 생성 flow가 수행된다.

Job Command Shared Memory :
2-5기본사항 Job Command Shared Memory 생성 flow가 수행된다.

[2] Condition Before Job

2-1. Rail Info Shared Memory 생성

Rail Info 공유메모리는 Rail 상에 있는 Object들의 정보를 저장하고 있으며 실시간으로 정보가 갱신되어 진다. 갱신되는 정보는 Job Control Process가 최적의 경로 탐색 및 Vehicle(Robot) 선택 그리고 Vehicle(Robot)의 기동여부를 파악하기 위해 참조되어진다. 참조되는 정보를 저장하는 공유메모리 객체는 시뮬레이션 시작 시 객체로 생성되어 메모리에 로딩된다. Rail Info에서 관리하는 정보는 Rail Number, Vehicle(Robot), EQ, Objects들이다. 이러한 정보는 모니터링 실행 시 맵 정보파일에서 읽어온 데이터를 가지고 초기화되어진다. 초기화되어진 자료는 Job Info에 입력되는 작업지시 목록을 수행함으로써 정보들이 갱신되어 진다. 이들 정보는 Vehicle (Robot) Thread에 의해서 수정되어 진다.

2-2. Vehicle(Robot) Info Shared Memory 생성

Vehicle(Robot) Info 공유메모리는 Rail 상에 있는 Vehicle(Robot)들의 정보를 저장하고 있으며 실시간으로 정보가 갱신되어 진다. 갱신되는 정보는 Job Control Process가 최적의 경로 탐색 및 Vehicle(Robot)선택 그리고 Vehicle(Robot)의 기동여부를 파악하기 위해 참조되어진다. 참조되는 정보를 저장하는 공유메모리 객체는 시뮬레이션 시작 시 객체로 생성되어 메모리에 로딩된다. Vehicle (Robot) Info에서 관리하는 정보는 Rail Number, Vehicle(Robot), Objects, Status, 출발지, 목적지, 좌표들이다. 이러한 정보는 모니터링 실행 시 맵 정보파일에서 읽어온 데이터를 가지고 초기화되어진다. 초기화되어진 자료는 Job Info에 입력되는 작업지시목록을 수행함으로써 정보들이 갱신되어 진다. 이들 정보는 Vehicle(Robot) Thread에 의해서 수정되어 진다.

2-3. Object Info Shared Memory 생성

Object Info 공유메모리는 Rail 상에 있는 Object 들의 정보를 저장하고 있다. 참조되는 정보를 저장하는 공유메모리 객체는 시뮬레이션 시작 시 객체로 생성되어 메모리에 로딩된다. Object Info 공유메모리에서 관리하는 정보는 객체이름, 객체의 좌표, 다음에 오는 객체이름,초, 길이 등이다. 이러한 정보는 시뮬레이션 실행 시 맵 정보파일에서 읽어온 데이터를 가지고 초기화되어진다.

2-4. Job Info Shared Memory 생성

Job Info 공유메모리는 시뮬레이션 시에 작업지시 명령을 저장하는 메모리이다. 시뮬레이션 실행 전 Job Command 공유메모리를 생성한다. 모니터링에서 MCS와 통신하여 MCS로부터 작업명령을 받아서 Job Info공유메모리에 저장한다. Job Info 공유메모리에 저장된 작업명령은 Job Control Process가 참조한다. Job Info 공유 메모리에 저장되는 정보는 Job ID,목적지, 출발지 정보를 이다.

2-5. Job Command Shared Memory 생성

Job Command 공유메모리는 시뮬레이션 시에 작업지시 명령을 저장하는 메모리이다. 시뮬레이션 실행 전 Job Command 공유 메모리를 생성한다. Job Info 공유메모리에 작업 명령이 도착하면 Job Control Process가 작업명령을 읽어서 최적의 Vehicle(Robot)과 작업경로를 선택하여 작업명령을 Job Command에 저장한다. 저장된 Job Command가 실행을 위해서 Vehicle(Robot) Thread가 생성되어지고 생성된 Vehicle(Robot) Thread가 작업을 수행한다.

[3] Basic Flow of Events

이 유스 케이스는 Job Control에 의해서 주어진 경로를 따라 움직이는 Vehicle(Robot)을 시뮬레이션 하는 것이다. 작업명령이 주어지면 다음의 과정을 거쳐서 시뮬레이션을 한다.

3-1 VH Info에서 작업 명령을 할당받은 VH의 정보를 찾는다.
3-2 VH의 정보를 가지고서 OBJINFO에서 OBJ 시작 정보를 찾는다.
3-3 OBJ INFO 에서 시작점이 목적지인지 판단한다.
3-4 목적지이면 움직이지 않는다.
3-5 목적지가 아니면 Vehicle(Robot) 의 경로방향을 결정한다.
3-6 목적지까지 1씩 증가한다.
3-7 목적지까지 경로를 표시한다.
3-8 위치 값이 바뀌면 이벤트를 발생시킨다.
3-9 Main Thread는 이벤트를 받아서 경로를 따라서 Vehicle(Robot)을 그린다.
3-10 Vehicle(Robot)을 화면에 그릴 때 Vehicle(Robot)의 현재명령 모드를 참조하여 그린다.

[4] Alternative Flow of Events

[5] Special Requirements
N/A

[6] Preconditions

[7] Postconditions
N/A

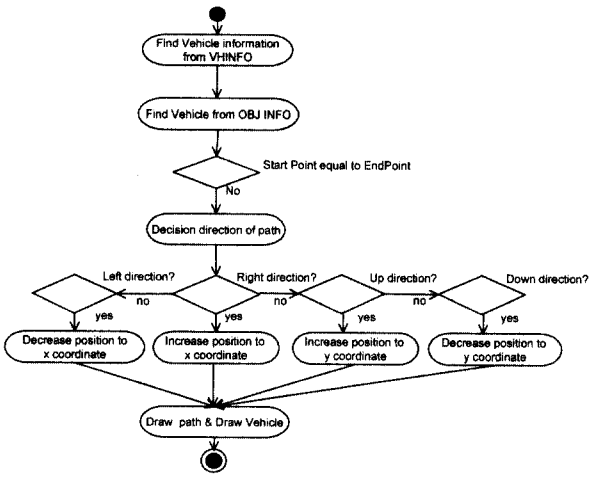
[8] Pictures of the User Interface

3.3.3 유스케이스 이벤트 흐름

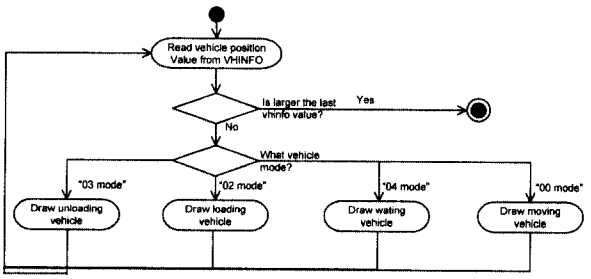
유스 케이스 이벤트 흐름을 기술하는 방법은 <표 4>와 같이 서술형으로 기술하는 방법과 의사 코드(Pseudo Code)를 이용하는 방법, 다이어그램 혹은 순서도를 이용하여 표기하기도 한다. 각각의 방법이 모두 장단점을 가지고 있으며 어떤 방법을 통하여 이벤트의 흐름을 표시해야 하는가는 중요하지 않다. 다만 어떤 방법을 사용하던지 유스 케이스의 모든 이벤트 흐름을 찾아내는 것과 작성된 이벤트 흐름이 사용자 및 개발자의 관점에서 모두 이해가 용이하도록 되어야 한다. 각각의 유스 케이스 이벤트 흐름을 표시하기 위해서 활동도(activity diagram)을 사용한다. 활동도는 이벤트의 흐름을 각각의 동작상태(action state)로 구분하고 동작상태 간의 변화(transition)를 다이어그램으로 표현한다.

동작상태는 이벤트의 흐름에서 발생되는 활동 혹은 단계들을 표현하고, 이러한 동작 상태의 변화를 화살표를 사용하여 표현함으로서 다이어그램을 완성한다. 이때 각각의 상태 변화를 표현할 때 조건(guard condition)을 줄 수 있으며, 이러한 조건들에 대한 분기가 발생하는 경우 분기 표시도 가능하다. 이때 병렬 처리되는 이벤트 흐름에 대한 부분도 표현 가능하다. (그림 5)는 Simulation Robot Job에 대한 이벤트 흐름을 활동도로 표현한 것이다.

(그림 5)는 (그림 3)에서 로봇이 경로를 따라서 이동 할 경우, 이동 방향에 따른 위치 값을 갱신시키면서 로봇의 위치를 사용자 화면에 그려주는 활동도이다. (그림 6)은 로봇의 작업 상태를 파악하여 작업상태에 따른 로봇의 모양을 화면에 그려주는 활동도이다. 로봇의 작업모드가 02이면 웨이퍼를 로딩하는 경우로써 로봇의 색깔을 빨간색으로 그려준다. 로봇의 작업모드가 03이면 웨이퍼를 언로딩하는 경우로써 로봇의 색깔을 연두색으로 그려준다.



(그림 5) 로봇의 이동경로에 따른 모니터링 활동도

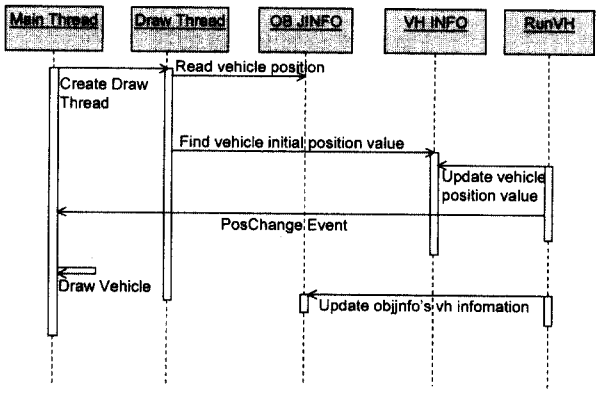


(그림 6) 로봇의 작업상태에 따른 모니터링 활동도

3.4 시퀀스 다이어그램

UML에서 객체를 찾아내고 객체간의 메시지를 파악하는 Use Case 실체화는 인터랙션 다이어그램으로 표현한다. 인터랙션 다이어그램은 시퀀스 다이어그램과 콜러보레이션 다이어그램이 있다. 시퀀스 다이어그램은 작성된 시나리오로부터 객체를 찾아내며 객체와 객체가 주고 받는 메시지를 파악하여 진행순서에 따라 표현한 것이다. 콜러보레이션 다이어그램은 객체와 객체사이의 연결과 메시지 자체에 초점을 두고 작성되어 진다.

시퀀스 다이어그램은 액터의 인스턴스와 각각의 식별자를 갖는 객체 그리고 그들이 주고받는 메시지로 이루어진다. 시퀀스 다이어그램을 처음으로 작성하는 경우에는 모든 객체에 대한 클래스가 아직 지정되지 않았으며 이후 작업을 통하여 클래스가 추출되면 해당 객체에 대해서 클래스를 지정한다. 이러한 작업의 반복을 통하여 업무영역에 존재하는 모든 객체와 모든 클래스를 파악하게 된다.



(그림 7) 시퀀스 다이어그램

(그림 7)은 메인 스레드에서 모니터링과 시뮬레이션을 위한 드로우 스레드를 생성한다. 생성된 드로우 스레드는 로봇의 위치정보를 OBJINFO 메모리에서 읽어온다. 그리고 VHINFO 메모리에서 드로우 스레드가 로봇의 초기 위치정보를 찾는다. 이때 VH16 INFO 메모리에는 RUN VH 스레드에 의해서 로봇의 최신정보가 갱신되어 진다. 이때 VHINFO 스레드가 위치변화 이벤트를 발생 시켜 메인 스레드에게 알려주면 메인 스레드는 로봇의 위치를 사용자 화면에 그려준다.

3.5 클래스 설계

요구분석 단계에서 유스 케이스 모델과 유스 케이스 스펙을 통해서 객체들과 객체들간의 메시지와 이벤트를 파악한 것을 바탕으로 객체와 클래스를 추출하고 클래스들 간의 관계성을 정의하는 클래스 분석단계를 진행한다.

3.5.1 클래스 정의서

클래스 정의서는 클래스 설계과정에서 만들어진 클래스 목록이다. 그리고 클래스를 식별하는 작업은 많은 응용영역들에 대한 경험이 필요하지만, 클래스를 식별할 때 다음과 같은 기준을 적용한다.

기준 1 : 저장하거나 분석할 데이터를 찾는다. 이런 데이터는 시스템에 적용해야 하는 개념이나 특정한 순간

에 발생하는 이벤트 혹은 트랜잭션으로 간주된다.

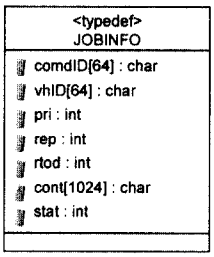
기준 2 : 외부 시스템을 찾는다. 외부 시스템은 개발중인 시스템이 상호작용 하여야 하는 클래스로 간주 할 수 있다.

기준 3 : 패턴, 클래스 라이브러리, 컴포넌트를 찾는다.

기준 4 : 액터가 어떤 역할을 수행하는지 찾는다. 사용자, 시스템 오퍼레이터, 고객 등과 같은 액터의 역할은 클래스로 간주될 수 있다.

• JOB INFO CLASS

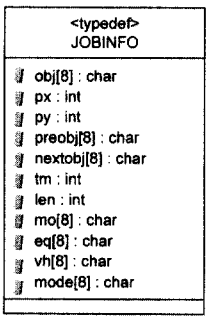
Job Info 구조체는 호스트로부터 수신 받은 job 정보를 저장하는 구조체이다(그림 8). 여기에 있는 필드는 여러 가지 정보를 저장한다. 여러 가지 정보들은 ^와 # 같은 구분자에 의해서 분리되어진다.



(그림 8) Job Info Class

• OBJ INFO CLASS

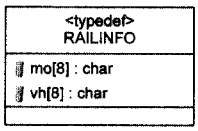
이 구조체는 object 정보를 저장한다. Object 정보는 레일이나 stoker 정보이며 또한 Vehicle(Robot)과 equipment 정보를 저장한다(그림 9).



(그림 9) Obj Info Class

• RAIL INF CLASS

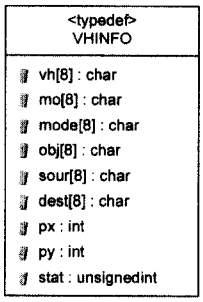
Rail info 구조체로서 rail의 정보를 저장하기 위해서 사용된다(그림 10).



(그림 10) Rail Info Class

• VH INFO CLASS

Vehicle(Robot) 정보를 저장하기 위한 구조체이다. 상태필드는 Vehicle(Robot)의 상태를 저장한다. (그림 11)은 Vehicle(Robot)의 상태도이다.



(그림 11) Robot Info Class

3.5.2 스트레오 타입

스트레오 타입은 UML의 어휘를 확장하는 것으로 기존의 UML 표현 양식에 근거하여 새로운 유형의 표현 양식을 정의하는 것이다. 사용자는 스테레오 타입을 이용하여 자신의 문제에 적합한 유형의 새로운 표현 양식을 작성할 수 있다[10]. <표 4>는 본 시스템에서 사용된 스트레오 타입의 목록들이다.

3.5.3 클래스 다이어그램

클래스가 추출된 후 각 클래스에 대한 관계를 설정하며 각 클래스간의 관계 설정과 관련된 여러 가지 디자인 패턴을 적용한다. 기본적인 클래스 다이어그램간의 관계는 다음 세 가지로 분류되며 이 세 가지 관계를 기본으로 하여 클래스 다이어그램을 작성했다.

〈표 5〉 사용된 스트레오 타입

Stereo Type Name	escription	Rose Icon if Used
Typedef	This stereotype represent the a C/C++ structure. The definition of the structure is given as attributes of the stereotype class.	No Special icon is used in the diagram
Shared Momory	This stereo type represent a structure or array which is a shared in the memory between processes. This is not a C++ class. The name of shared memory is stereotype object. An it's type detail is Described as the attribute detail.	
Global	This stereotype is used to represent the global variables and global functions, This is not a C++ class, The global stereotypes name is the application s name. And the gobal attributes are represented as the attributes and the global functions are given as the operations of the stereotype class.	

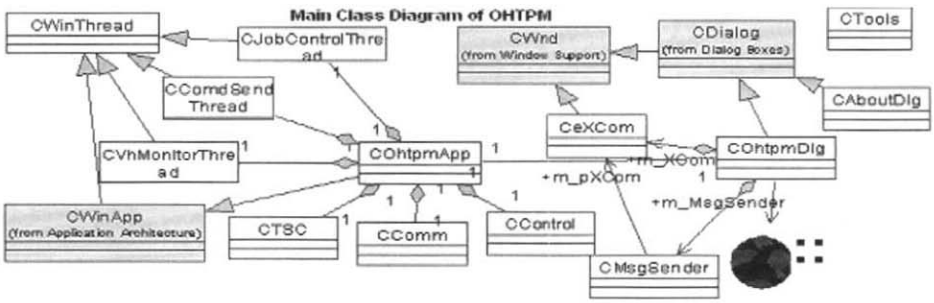
① 종속 : 종속은 클래스간의 사용관계를 나타낸다. 즉, 한 클래스가 다른 클래스를 사용할 경우 종속을 사용한다. 종속은 사용되는 클래스가 변경되면 클래스의 인터페이스나 행위가 달라질 수 있기 때문에 이를 사용하는 클래스가 영향을 받는다. 종속의 가장 일반적인 유형은 오퍼레이션의 매개변수로서 다른 클래스를 사용하는 클래스간의 연결이다. 이러한 사용관계를 모델링하기 위해서 오퍼레이션을 갖는 클래스로부터 오퍼레이션의 매개변수로 사용되는 클래스로 향하는 종속을 생성하여야 한다.

② 일반화 : 일반화는 수퍼클래스와 보다 세부적인 서브클래스 간의 관계이다. 일반화는 "is-a-kind-of" 관계라고도 한다. 서브 클래스는 속성이나 오퍼레이션 등과 같은 부모의 특성을 상속받는다. 부모, 자식 관계를 표현할 때 일반화를 사용한다. 일반화는 대부분 상속관계를 표현하기 위해 클래스와 인터페이스간에 사용된다.

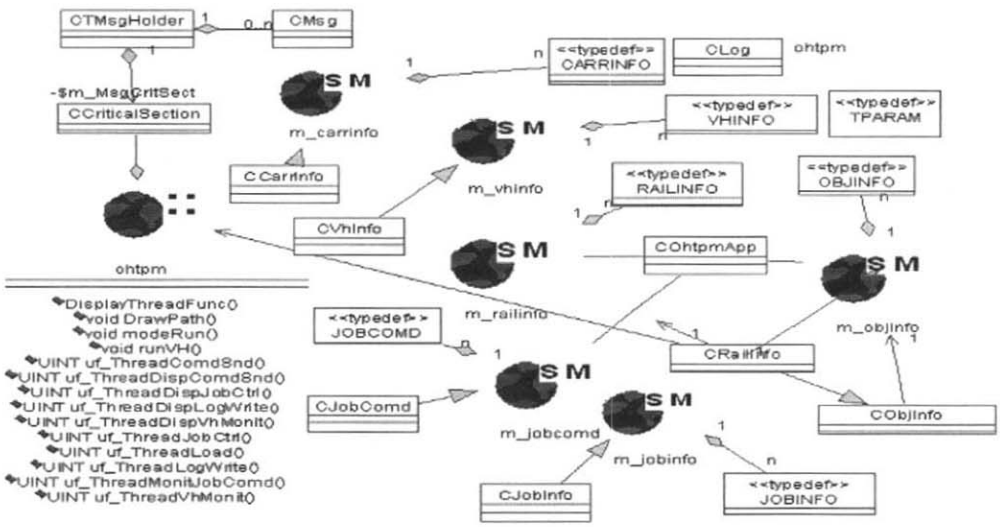
③ 연관 : 연관은 객체와 객체를 연결하는 구조적이 관계를 의미한다. 연관은 이름, 역할, 다중성, 집단화 등고 같은 네 가지의 추가적인 표현 사항을 갖는다. 두 개의 클래스를 연결하는 연관에서 한 클래스의 객체로부터 다른 클래스의 객체로 네비게이션을 할 수 있다. 클래스가 연관 관계를 가질 때 각각의 클래스는 각자의 특정한 역할을 갖는다. 역할은 연관에 참여하는 하나의 클래스가 다른 클래스에게 제공하는 책임과 같다. 다양한 모델링 상황에서 객체가 얼마나 많은 연관에 연결되어 있는가를 표현할 수 있다. 이를 다중성(multiplicity)이라고 한다.

일반적인 두 클래스간의 연관은 동등한 수준에서 두 클래스간의 구조적인 관계를 표현한다. 그러나 때로는 하나의 클래스는 보다 큰 집합을 표현하고 다른 클래스는 보다 작은 집합으로 표현할 경우가 있다. 이러한 관계유형을 집단화(aggregation)라고 하며, 전체에 속하는 객체가 부분에 속하는 객체를 갖는 "has-a"의 관계를 나타낸다.

(그림 12)는 시스템의 메인 클래스 다이어그램이다. 메인 클래스인 COhtpmApp는 상위 시스템으로부터 작업명령을 받는다. 이때 작업 명령을 받기 위한 프로토콜로 IBSEMI 표준을 만족하는 SECSII 사용한다. 작업 명령이 떨어지면 메인 클래스는 Job Control Thread를 생성하여 로봇의 위치를 추적하고 로봇에게 작업명령을 할당하며 로봇의 현재



(그림 12) 시스템의 메인 클래스 다이어그램



(그림 13) 시스템의 공유메모리 클래스 다이어그램

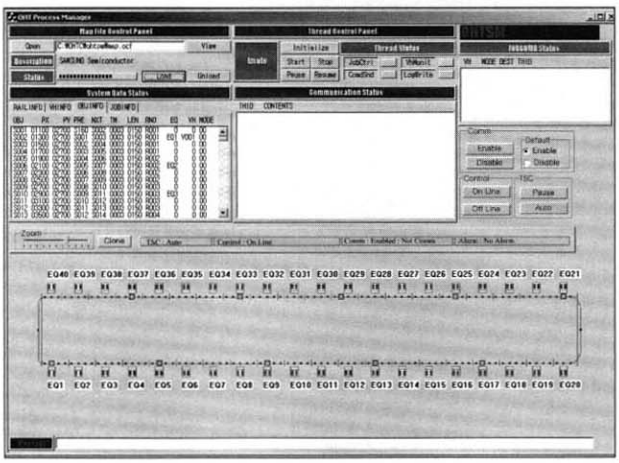
상태를 모니터링 한다.

(그림 13)은 이 시스템에서 사용하는 메모리 클래스의 관계를 보여주고 있다. 즉 로봇의 현재정보, 작업정보, 그리고 맵의 정보를 실시간으로 저장하는 메모리들 사이의 관계를 클래스 다이어그램으로 보여주고 있다.

3.6 사용자 인터페이스

사용자 인터페이스는 (그림 14)와 같이 구성되어 있다. 맵 파일 제어부는 반도체 생산라인 공정에서 필요한 로봇의 개수와 웨이퍼를 올리고 내리는 장비 개수 등 여러 가지 맵 정보를 가지고 있는 파일을 관리하는 영역이다. 이 맵 파일을 열어서 로딩하면 시스템 데이터 모니터링 영역에 데이터가 기록된다. 스레드 제어부는 맵 파일에서 기본 정보를 읽은 후 작업을 수행하기 위한 스레드를 관리하는 영역이다. MCS 통신 모니터링 영역은 IBSEMI 프로토콜을 통하여 MCS로부터 전송되는 작업 명령을 모니터링 하는 영역이다. 작업상태 모니터링 영역은 현재 로봇의 작업 모드를 모니터링 한다. 모니터링 맵 축소/확대 영역은 맵 디스플레이 영역을 확대/축소하는 기능이다. 맵 디스플레이 영역은 스레드 제어부에 의해서 실행되는 드로우 스레드가 시스템 데이터 모니터링에서 로봇의 데이터를 읽어서 화면에 보여주는 영역이다. 실제로 모니터링 되면서 로봇이 화면상에서 움직이는 영역이다. 메시지 출력부는 작업과정에서 필요한 정보를 화면에 출력하는 영역이다.

(그림 14) 사용자 인터페이스 구성도



(그림 15) 사용자 인터페이스 화면

그리고 실제로 구현된 사용자 인터페이스 화면은 (그림 15)와 같다.

4. 결 론

정보 시스템들이 복잡화, 대형화 통합화로 발전하고 있다. 소프트웨어 개발에서도 이러한 현실을 반영하기 위해서는 사용자의 요구사항들을 보다 세밀하게 분석하고, 설계를 통한 전체적인 개발 공정관리와 체계성을 필요로 하게 되었다. 이러한 필요조건을 충족하기 위한 개발 방법론으로 UML을 사용하여 반도체 웨이퍼 생산 공정에서 필요한 로봇을 모니터링하고 시뮬레이션 하기 위한 시스템을 설계 및 구현하는데 적용하였다.

이러한 개발 방법론을 적용함으로써 고객과의 의사소통이 잘 이루어지고, 고객의 요구사항을 더욱 더 세분화시킬 수 있었다. 또한 시스템 개발 과정에서 피드백을 통한 고객의 요구사항이 원활하게 반영되었다. 또한, 소프트웨어의 모듈단위의 설계가 잘 이루어졌다.

이 시스템에서 앞으로 개선되어야 할 부분은 로봇의 작업 경로를 최적화하기 위한 알고리즘 개발과 작업 경로의 최적화 알고리즘을 개발한 후 이 알고리즘 모듈이 이 논문에서 기술한 시스템에 잘 이식되어 질 수 있도록 하는 것이다.

참 고 문 헌

[1] OMG, UML, Specification version 1.1, Object Management Group, November, 1997.

[2] Martin Fowler, "UML DISTILLED : Applying the Standard Object Modeling Language," Addison-Wesley, 1997.

[3] Craig Larman, "Applying UML and Patterns — An Introducation to Object-Oriented Analysis and Design," Prentice Hall, 1998.

[4] 장옥배, 유철중, 이병걸, 김지홍, 양해술, 김병기, "소프트웨어 공학", 도서출판 한산, 2001.

[5] 류형규, 이순천, 류시원, 신성호, "UML 기반 객체지향 클라이언트/서버 구축", 홍릉과학출판사, 2000.

[6] Doug Rosenberg, Kendall Scott, Use Case Driven Object Modeling with UML : A Practical Approach, Addison-Wesley, 1999.

[7] Hans-Erik Eriksson, Magnus Penker, UML Toolkit, Wiley Computer Publishing, 1998.

[8] Martin Fowler, Kendall Scott, UML Distilled Second Edition : A Brief Guide to the Standard Object Modeling Language, Addison-Wesley, 2000.

[9] Wirfs-Brock, R., et al., Designing Object-Oriented Software, Prentice Hall, 1990.

[10] 조완수, "UML객체지향 분석설계", 홍릉과학출판사, 2000.

심 갑 식

e-mail : gssim@jinju.ac.kr
1987년 전남대학교 계산통계학과(이학석사)
1993년 전남대학교 전산통계학과(이학박사)
1993년~현재 진주산업대학교 교양학부 부교수
관심분야 : 정보보안, 무선인터넷, 데이터 모델링, XML

정 태 영

e-mail : garisan@lycos.co.kr
1996년 경상대학교 전자재료공학과(공학사)
1997년 경상대학교 전자재료공학과 (공학석사)
1997년 (주)광성전자 선임연구원
2003년 에이스코리아 대표
관심분야 : 데이터마이닝, CRM, 무선인터넷