FMC 제어 소프트웨어의 객체지향적 설계
- Object-oriented design of FMC Control Software -

조 용 탁*
Cho, Yong-Tak
한 영 근**
Han, Young-Geun

Abstract

As a software development methodology, object-oriented paradigm that has execellent reusability, portability, and extensibility, is currently being used in many application fields. Especially, UML(Unified Modeling Language), which is recently released as a third generation methodology for the object-oriented system development, has many advantages such as generalization, certainty, visualization. For this reason, the usability of UML in manufacturing system control is expected to increase. In this paper, analysis and design of FMC control system are performed by UML. Software objects to accommodate the dynamic environment of FMC operations are modeled by using the diagrams of UML. The objective of this paper is to suggest a generic framework to design FMC control software

1. 서론

FMC(Flexible manufacturing Cells)는 유사한 가공 공정을 갖는 다양한 부품을 유연하게 제조하기 위해 설치된 자동화된 셀이라 할 수 있다. 최근들어 고객의 제품에 대한 요구사항이 다양해짐에 따라, 혹은 FMS(Flexible Manufacturing Systems)로의 발전을 위한 기초로서 FMC의 중요성을 평가할 수 있다. 이러한 FMC를 활용하는데 있어 셀 제어는 중요한 문제 중에 하나이다. 본 연구에서는 셀을 유연하게 제어할 수 있는 소프트웨어를 신속하게 구현하기 위한 방법에 관하여 논의한다.

제어 소프트웨어를 디자인할 때 과거의 구조적 분석 방법을 이용할 수도 있지만, 시스템을 분석에서부터 구현에 이르기까지 품질을 향상시키고, 개발 비용과 시간을 감소시키기 위해서 많은 개발자들은 더욱 효과적인 방법을 찾게 되었는데 재사용성, 이식성, 확장성등이 우수한 객체지향 방법이 그 중의 하나이다..

그러나 객체지향 연구자마다 표기법과 적용 분야, 적용 시스템 규모를 달리하고 있어 이 분야는 심지어 "method wars" 상태에 있다고 말해지기도 한다.[7] 즉 동일한 시스템을 디자인했을 지라도 개발자마다 각기 다른 디자인을 하게 되며, 이로 인하여 시스템 확장 및 시스템 구성 변경에

* 명지대학교 산업공학과 석사과정
** 명지대학교 산업공학과 조교수

따른 시스템 재디자인이 필요할 때 많은 비용 및 시간이 소요된다. 본 연구에서는 여러 연구자들이 개발한 객체지향 방법을 통합한 UML을 이용하여 정형화된 표기법에 따른 FMC 제어 소프트웨어 시스템 디자인을 시도한다.

2. FMC 제어

생산 시스템의 제어는 시각에 따라 다양하게 정의될 수 있으며, 셀 제어를 위해서는 셀 일정계획(cell scheduling), 셀 모니터링(cell monitoring), 셀 조정(cell coordinator), 셀 통제(cell control), 셀 통신(cell communicator)과 같은 기본적인 요소들이 필요하다.[2] 본 연구에서의 셀 제어는 장비 셋업, 장비의 작동 및 일시정지, 작업을 수행하기 위한 명령을 내보내기 등과 같은 일련의 operation들을 운용하는 것을 말한다. 이러한 제어 기능을 위한 구조는 일반적으로 상하 관계가 존재하는 Hierarchical 구조가 많이 이용된다. 즉 그림 1과 같이 각 장비의 제어기 상부에 그 셀의 제어기가 존재하고 또 그 상부에 전체 시스템의 제어기가 존재하여 명령은 상에서 하로, 상태 보고는 하에서 상으로 이동하여 각 레벨에 따라 분담된 제어 기능을 수행하게 된다.

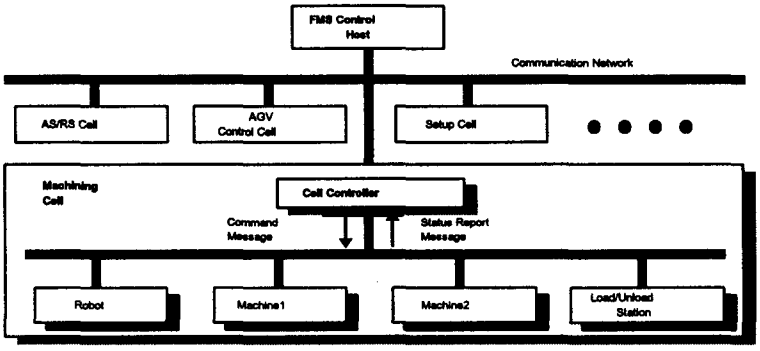

그림 1. 셀 제어 구조

FMC 제어 시스템은 실시간의 현상에 응답할 수 있어야 하는데, 이러한 응답 매커니즘을 UML을 통해 소프트웨어로 구현하기가 용이하다. UML이 제공하는 diagram들을 이용하여 FMC 운영을 모델링하기 위해 필요한 소프트웨어 객체들을 표현한다. 개개 객체가 자신의 활동을 책임지는 반면에 셀 제어기는 객체들간의 조정 및 동시화 문제를 처리한다.

3. 객체지향 설계

3.1 객체지향 paradigm

객체지향 paradigm이란 "상호작용하는 객체들의 집합"으로 시스템을 다루는 방법이라고 할 수 있다. 이를 바탕으로 객체지향 방법은 객체 및 그 구조를 정의하는 정적 관계(Static Relation) 부분과 객체들간의 상호작용을 정의하는 동적 행위 (Dynamic Behavior) 부분으로 소프트웨어를 파악한다.[3]

3.2 기존의 객체지향적 설계 기법

여러 연구자들이 각자의 객체지향 방법론을 제시하였는데 그것들을 간략히 열거하면 다음과 같다.

Jacobson의 OOSE(Object Oriented Software Engineering)는 Use Case를 가장 특징적인 개념으로 볼 수 있는데, 이는 시스템 외부와 시스템이 어떻게 상호 작용하는지를 파악함으로써 시스템의 기능을 설명하는 것이다.[3]

Rumbaugh의 OMT(Object Modeling Technique)를 이용한 객체지향 개발 방법은 주로 응용영역 개념(Application-domain Concept)을 중심으로 기술되었다.[3][4]

Booch의 방법은 Ada를 이용한 소프트웨어 개발에 초점을 두었으므로 객체와 클래스, 객체간의 상호 작용에 대한 풍부한 개념을 지원하고 있다. 이 방법은 실시간 시스템이나 병행성을 지닌 시스템을 모형화하기에 가장 바람직하다.[3][4]

Coad와 Yourdon의 방법의 특징은 체계적인 방법이 아닌 대화적으로 개발을 진행해 나가는데 있다. 그러나 이 방법은 객체 간의 통신 지원 개념이 취약하고 대규모 프로젝트보다는 작은 시스템 개발에 적합하다는 단점이 있다.[3][4]

이 외에 Wirfs와 Brock, Martin과 Odell, Shlaer와 Mellor 등이 방법론을 제시하였다. 하지만 각 연구자들 마다 표기상의 차이가 있으며, 각기 다른 적용대상과 범위를 지니고 있다.

3.3 객체지향 기법의 생산시스템 응용 사례

Adiga와 Lin은 실제 객체와 그 객체의 제어 과정을 모델링했고, 동적 변경을 MFD (message flow diagram)을 이용하여 모델링하였다.[6][8]

Miller와 Lennox는 소프트웨어 객체 층과 함께 작동할 수 있는 로봇 제어기를 개발하였는데, 그 시스템은 유연성과 Fault Tolerance를 유지하면서 계층적 FMC 제어 시스템을 모델링한 것이다.[9]

국내에서도 유사 응용 연구가 이루어졌는데, Use Case Driven 기법을 이용한 다공장의 계층적 구조를 가진 생산계획 및 통제시스템에 대한 객체지향적 분석 및 설계를 수행한 사례와[2] FMS 제어 소프트웨어 구현을 위한 객체 지향형 FMS 데이터 모델을 제안하고 상업용 객체 지향 데이터베이스 시스템을 이용해 FMS 제어 소프트웨어 시스템을 구현한 사례를 들 수 있다.[10]

3.4 UML(Unified Modeling Language)

3.2에서 기술한 바와 같이 여러 방법들이 가지고 있는 불필요하거나 잘 사용되지 않는 모델들을 제거하고, 사용자들의 혼동 요인을 최소화시키려고 제시한 방법론이 UML이다. Booch, Jacobson, Rumbaugh에 의해 개발된 제 3세대 객체지향 방법론인 UML은 객체지향 시스템을 분석할 뿐만 아니라, 소프트웨어 개발 목적으로 시스템을 규정, 기록, 가시화, 구축하는 언어이며, 동시적, 분산적인 시스템을 모델링하는 기법도 제공한다. 향후 객체지향 시스템을 개발하기 위한 표준 모델링 방법으로 널리 사용될 것이 예상된다.

4. FMC 제어의 UML 설계 도구 적용

UML은 유용한 설계 도구를 제공하는데, 시스템의 분석 및 디자인 단계에 적합한 설계 도구와 실제로 제어 소프트웨어를 구현하는 단계에 적합한 설계 도구으로 나눌 수 있다. 전자에 속하는 것은 class diagram, use case diagram, state diagram, activity diagram, sequence diagram, collaboration diagram 이고, 후자에 속하는 것은 component diagram, deployment diagram이다.

본 연구에서는 시스템의 분석 및 디자인에 적합한 모델 중 class diagram, use case diagram, sequence diagram, state diagram을 이용하여 시스템을 모델링 한다. 적용할 다이아그램별로 그 특징을 살펴보면, class diagram은 FMC 시스템을 구성하는 클래스와 클래스간의 관계를 기존의 방식보다 더욱 상세하고 명확하게 보여 줄 수 있다. use case diagram은 FMC 시스템에서 이루어지는 상호작용을 사용범례를 중심으로 분석하므로 정확하게 분석할 수 있다. sequence diagram은

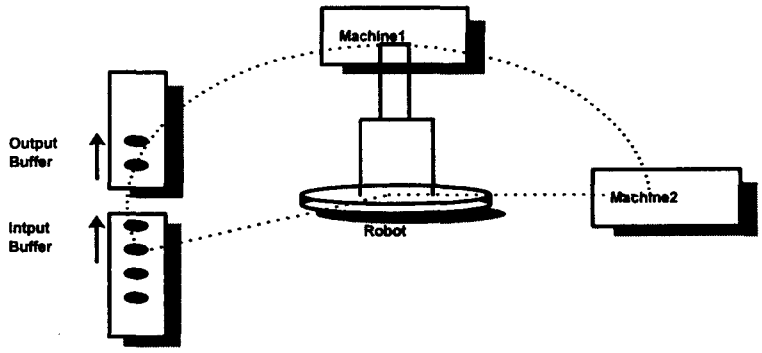

그림 2. 가상 FMC의 구성도

객체간에 교환되는 메시지를 시각에 따라 분석하므로 전체 시스템이 작동되는 과정을 쉽게 분석할 수 있다. state diagram은 클래스를 표현하는 방식으로 각 객체들이 갖을 수 있는 state를 표현하므로 복잡한 생산시스템의 state를 간단하면서도 정확히 표현할 수 있다. 적용될 가상 FMC 모델은 그림 2와 같이 1대의 material handling 로봇, 2대의 기계, 입력/출력 버퍼로 구성되어 있다.

4.1 Class Diagram

Class diagram은 모델의 class, type, class간의 내부 구조, 다른 class와의 관계와 같은 정적 구조를 보여준다. 시간 변화에 따른 정보는 보여주지 않지만, 시스템 개발 단계에 따라 압축된 클래스 형태, 분석 단계의 클래스 형태, 구현 단계의 클래스 형태로 클래스를 표기할 수도 있다.

그림 3은 분석 단계의 class diagram이다. class을 세 부분으로 나뉜 사각형으로 표기하며, 사각형의 맨 위부터 아래로 class name, attributes, operations을 기입한다. 독립적인 클래스들간의 의미적 연결을 association이라 한다. 각 클래스를 연결하는 선 위에 쓰인 동사 및 동사구는 association name을 나타내는데, association name 옆에 있는 작은 삼각형이 가리키는 방향으로 association name을 분석한다. 예를 들면, Machine class와 NC prog DB class와의 association 관계를 "Machine은 NC prog DB를 Use한다."로 해석할 수 있다. 연결선에 사용된 숫자 및 *는 클래스의 instance 개수를 의미한다. 또한 클래스들간의 관계는 association, generalization, composition 중 하나로 표현할 수 있는데 그림 4는 세가지 관계 중 composition을 이용한 클래스 관계 표현을 설명한 것이며, 세 가지 다른 형태로 표현될 수 있다. 상단부의 표현은 상위 클래스를 표시하는 마름모와 객체 instance를 나타내는 숫자 및 *을 이용하여 composition을 표현한 것이고, 하단부의 두 그림은 cell class내에 속해 있는 모든 객체들을 보여주는데 좌측 그림은 각 객체 안에 instance 개수를 나타냈고, 우측 그림은 각 객체와 상위 클래스를 연결하는 선 위에 instance 개수를 나타낸 것이다.

4.2 Use case diagram

시스템과 상호작용하는 것들을 actor라 하고 시스템내에서의 activities을 use case라 할 때, 한 개 이상의 외부 actor들과 use case들간의 관계를 보여주는 것이 use case diagram인데, 이는 시스템내 activities의 경계와 시스템의 상황을 가시화하는데 유용하다.

actors는 stickman으로 표기하고, use case는 타원으로 표기한다. 그림 5는 robot, input/output buffer, machine이라는 actor와 check_order, send_command, inquire_status,

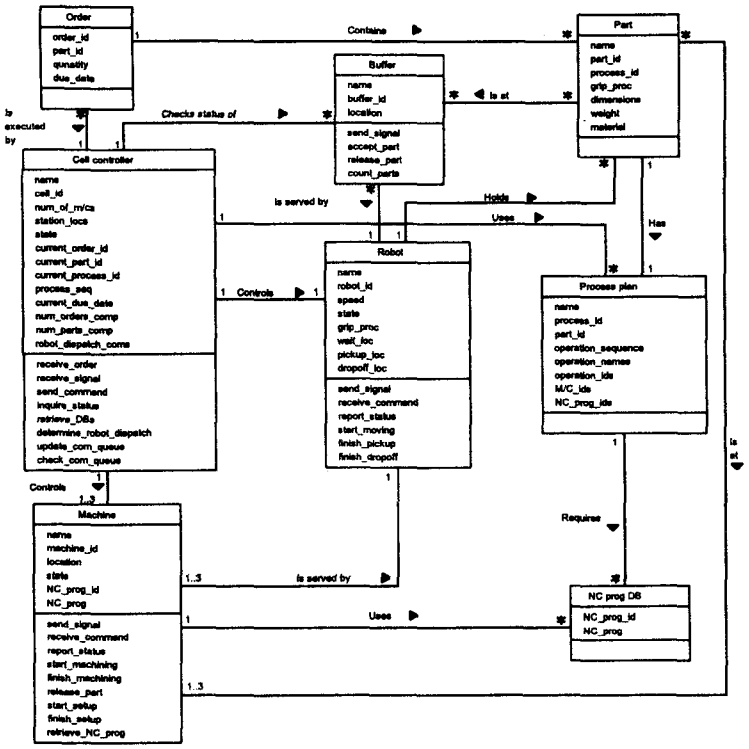

그림 3. FMC 제어의 class diagram

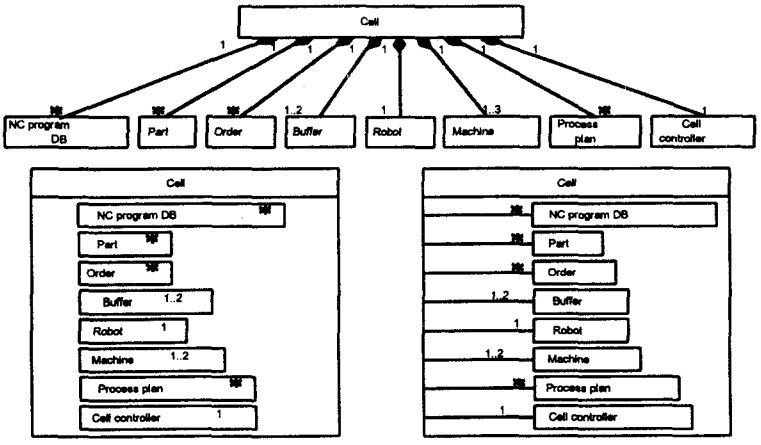

그림 4. Cell composition 표기법

determine_robot_ dispatch라는 use case들간의 상호작용 관계를 보여주고 있다.

4.3 Sequence diagram

Sequence diagram은 객체들 간의 상호작용 패턴을 메시지를 이용하여 시간적 순서에 따라 명확히 보여준다. 다이아그램에서 사각형은 class를 나타내고, class name에 밑줄이 그어진 것은

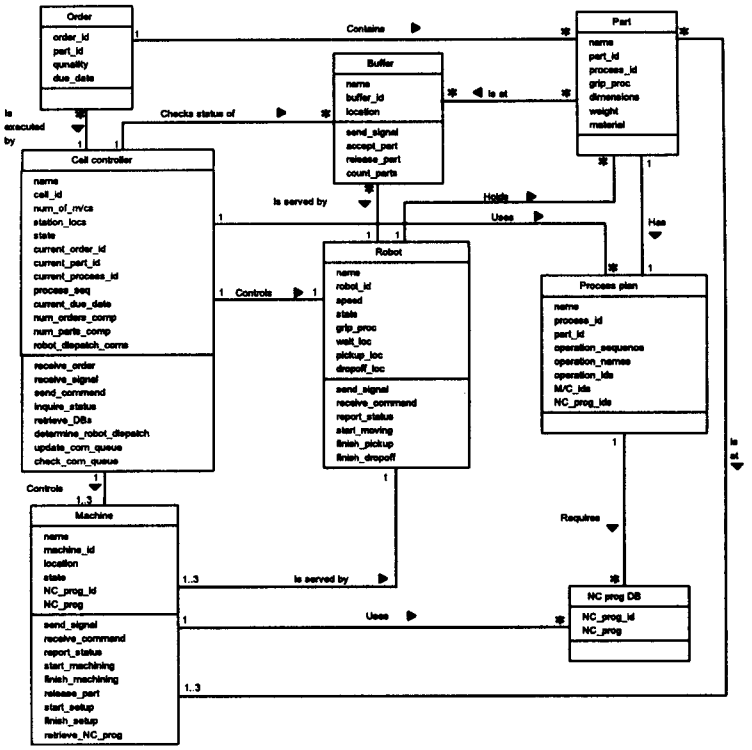

그림 3. FMC 제어의 class diagram

시스템을 구성하는 물리적 객체를 의미한다. 세로 막대 모양은 객체가 존재하는 기간을 나타내고, 세로 막대의 하단부로 갈수록 시간이 경과되는 것을 보여준다. 본 연구에서 로봇이 부품을 기계에

분배하는 과정에서 발생하는 Deadlock문제는 고려하지 않는다.

Input buffer에서 "전(前) 오더의 최종 부품이 나갔다."는 메시지가 셀 제어기에 보내지면 셀 제어기는 객체 데이터를 보내라는 메시지를 order, part, process plan 객체로 보내고 그 객체들로부터 객체 데이터를 받은 후, 기계 객체로 부터 "전(前) 오더의 최종 부품을 unloading하였다"는 메시지를 받으면 기계 객체로 새 오더에 해당되는 초기화 메시지를 보낸다. 셀 제어기는 robot에게 부품을 기계로 옮기라는 메시지를 보낸 후, 기계 객체로 가공하라는 메시지를 보낸다. 가공이 완료되면 기계 객체는 가공 완료 메시지를 셀 제어기에게 보낸다. 이와 같은 순서에 의해 한 오더에 대한 메시지 흐름을 그림 6과 같이 보여줄 수 있다.

4.4 State diagram

state diagram은 외부의 요청이나 메시지에 의해 반응하는 객체들과 이들의 상호작용이 취할 수 있는 state의 순서를 도시화한 것이다. state는 state name, state variable, internal activity로 구성되어 있다. state variable은 attribute와 같은 형태와 의미를 갖으며, internal activity는 객체가 임의의 state에 있는 동안에 수행할 수 있는 행동이다.

그림 7은 가상 FMC 제어기의 state diagram이다. 초기 상태는 오더 대기상태인데 오더를 받으면 다음 state로 바뀌기 전에 오더에 대한 정보가 있는 데이터베이스에서 각 객체에 대한 데이터를 읽어 들인다.

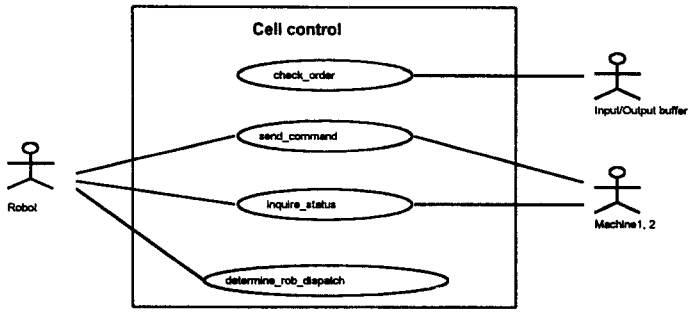

그림 5. FMC 제어의 use case diagram

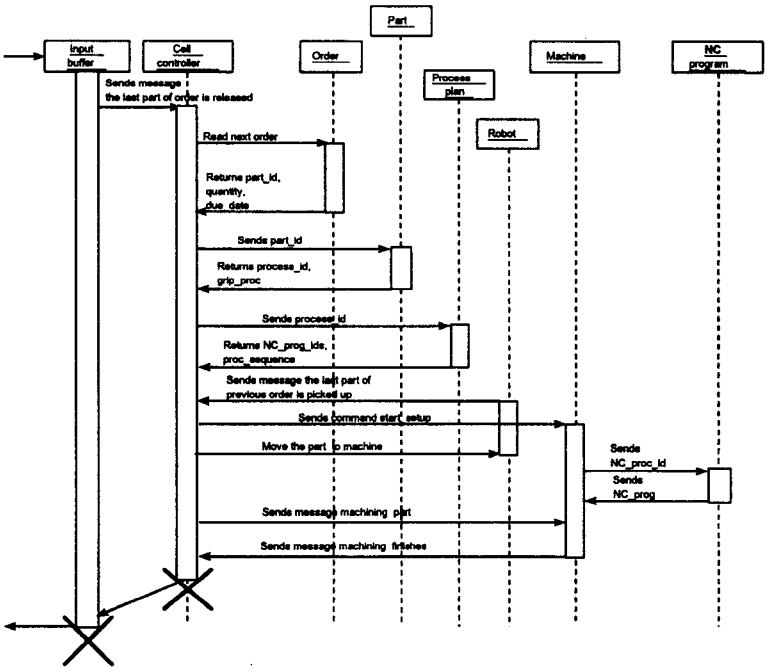

그림 6. FMC 제어의 sequence diagram

명령 큐는 셀 제어기에 있는 메모리 영역이며, 로봇 이동 명령이 FIFO의 순서로 대기하는 공간이다. 명령 큐를 점검하여 명령이 없으면 로봇이 쉬고 있는 신호대기 state로 바뀌고, 명령이 있으면 로봇이 이동하는 신호대기 state로 바뀐다. 이와 같이 state가 반복되다가 최종 오더의 마지막 부품을 output buffer에 놓았다는 메시지가 생성되면 셀 제어기는 초기상태로 된다.

4.5 UML 모델의 효과

UML은 기존의 객체지향 개발 방법들의 장점을 통합하면서도 복잡하거나 자주 사용되지 않는 표기는 제거하였기 때문에 시스템 분석 및 설계가 편리해졌으며, 시스템의 대규모화나 개발 분야의 다양화에도 일관된 방식으로 시스템을 쉽고, 정확하게 모델링할 수 있다. 또한 시스템 개발자들 측면에서 볼 때, 개발 방법의 변경에도 불구하고 UML 및 UML 전용 도구에 대한 많은 재교

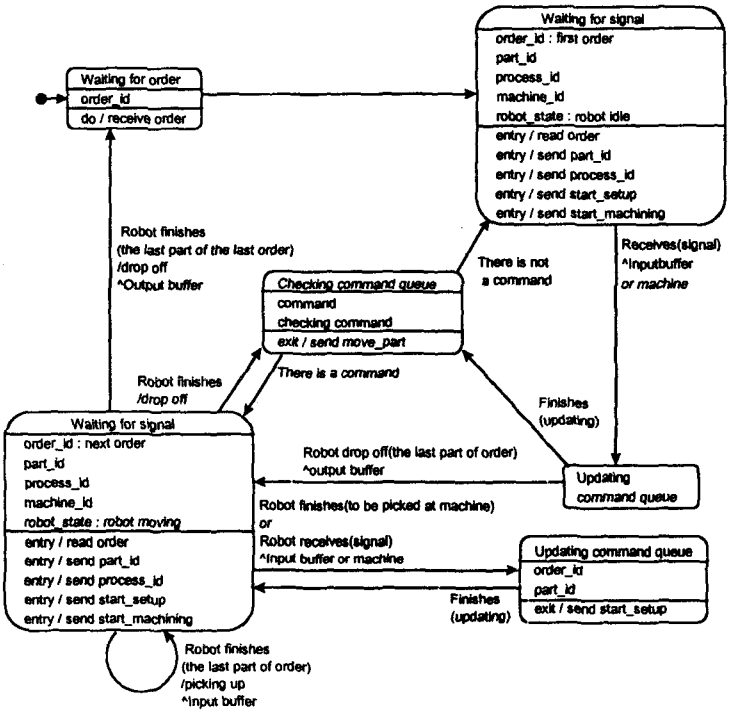

그림 7. FMC 제어의 state diagram

욱을 필요로 하지 않으며, 빠른 학습곡선을 기대할 수 있다.

Class diagram을 이용하여 각 객체간의 관계를 명확히 모델링 할 수 있고, sequence diagram과 state diagram을 이용하여 복잡한 시나리오를 가지며 실시간의 처리를 필요로 하는 생산시스템을 간단하고 명확하게 표현할 수 있다. 또한 use case diagram을 이용하여 시스템 분석시 생략되었거나 잘못 정의된 use case를 수정할 수 있다.

위와 같이, UML을 이용한 객체지향 모델링은 시스템 개발 단계(분석 및 설계, 구현)에 적합하게 클래스 및 객체 state를 표기 하므로 소프트웨어 개발의 자동화를 이루어 질적 향상 효과와 비용과 time-to-market 단축 효과를 얻을 수 있으며, 실시간 제어 기능을 요구하는 FMC control system을 간편하고 정확하게 모델링할 수 있도록 해 주며, FMS 및 CIM 시스템까지의 모델링 확장도 용이하게 해 준다.

5. 결론

본 연구에서 FMC 제어 소프트웨어의 설계 방법론을 제시하였다. FMC 제어 소프트웨어가 유연성을 갖도록 객체지향 paradigm을 적용한 UML을 이용하여 FMC 제어 기능을 형식 모델링 하였다. 이와 같은 방법을 이용하여 시스템 분석과 설계를 실시하고 시스템 제어 소프트웨어 구현에 적합한 모델을 구축함으로써, 생산시스템이 확장되거나 변경될 경우에 하드웨어와 소프트웨어를 민첩·유연하게 처리할 수 있다. 보다 다양한 제어구조에의 UML적용, 모델의 적합성 검증 및 제어 소프트웨어로의 구현 문제에 대한 연구가 추후 과제이다.

참 고 문 헌

[1] 강용혁, 서동욱, 김성식, "Object-Oriented Analysis and Design for A Hierarchical Planning and Control System of Inter-plant Production," '96 대한산업공학회 춘계공동학술대회 논문집, pp.204-208, 1996.

[2] 이경석, "FMS 분산제어 소프트웨어 구현을 위한 객체지향 framework의 개발", 석사학위논문, 서울대학교 대학원 산업공학과, pp.20-26, 1995.

[3] 최성운, 도홍석, 계원경, "객체지향 소프트웨어 개발 방법론 동향", 정보과학회지, Vol.14, No.10, pp4-11, 1996.

[4] 최영근, 허계법, 객체지향 소프트웨어 공학, 한국 실리콘, pp.89-169, 1995.

[5] Adiga, S., Gadre, M., "Object-oriented software modeling of a flexible manufacturing system," *J. Intell. Robotic Systems* Vol. 3, 147-165, 1990

[6] Adiga, S., *Object-oriented software for manufacturing systems*, Chapman & Hall, 1993

[7] Booch, G., Jacobson, I., and Rumbaugh, J. *"The Unified Modeling Language for object-oriented development,"* Documentation Set Version 1.0, January 1997.

[8] Lin, L., Masatoshi W., and Adiga, S., "Object-Oriented Modeling and Implementation of Control Software for A Robotic Flexible Manufacturing Cell," *Robotics & Computer-Integrated Manufacturing*, Vol. 11, No. 1, pp.1-12, 1994.

[9] Miller, D. J., Lennox, R. C. "An object-oriented environment for robot system architectures", *IEEE Control Systems,* Vol. 11, No. 2, pp.14-23, 1991.

[10] Sang K. Cha and Jang H. Park, "An object-oriented model for FMS control", *Journal of Intelligent Manufacturing*, Vol. 7, pp.387-391, 1996.