컨테이너크레인의 개방형 하이브리드 제어시스템에 대한 연구

홍경태* · 홍금식**

*부산대학교 대학원, **부산대학교 기계공학부 교수

Open-Architecture Hybrid Control System for Automatic Container Crane

*Kyung-Tae Hong** · *Keum-Shik Hong***

**Graduate school of Pusan National University, Pusan 609-735, Korea*

***School of Mechanical Eng., Pusan National University, Pusan 609-735, Korea*

요 약 : 본 논문에서는, 컨테이너 터미널에서 운행되는 컨테이너크레인의 자동화 시스템을 개방형 제어시스템으로 제안한다. 그리고, 개방형 제어시스템을 하드웨어 모듈과 OS 모듈 그리고 응용소프트웨어 모듈로 구성되는 표준모델로 제시한다. 본 논문의 기여도는 다음과 같다. 첫째, 크레인 제어시스템의 개방화를 위한 새로운 기준모델을 제안한다. 둘째로, 컨테이너크레인의 구조를 분석하고 자동화된 컨테이너크레인을 구현하기 위한 방법을 제안한다.

핵심용어 : 자동화 크레인, PC기반 제어, 개방형구조, 하이브리드시스템, 갠트리크레인, 제어시스템

ABSTRACT : *In this paper, an open architecture control system for automatic container cranes is investigated. A standard reference model for cranes, which consists of three modules; hardware module, operating system module, and application software module, is proposed. A hybrid control architecture combining deliberative and reactive controls for the autonomous operation of the cranes is proposed. The main contributions of this paper are as follows: First, a new reference platform for the crane control system is proposed. Second, by analyzing the structure of a container crane, implementation strategies for the automatic container crane are described.*

KEY WORDS : *automatic crane, PC-based control, open architecture, hybrid system, gantry crane, control system*

1. 서 론

국내 수출입 화물의 대부분이 항만을 이용하여 이루어지면서 항만을 통한 물류시스템은 그 비용이 연간 20조원에 이르는 국가기간산업으로써의 확고한 위치를 차지하고 있다. 현대의 항만 물류시스템은 글로벌 거점의 자동화시스템으로 전환되고 있으며, 얼마나 많은 양의 컨테이너를 얼마나 빠르게 취급할 수 있느냐에 따라 그 존폐가 좌우되고 있다. 따라서 항만에서의 최대관건은 주어진 시간에 얼마나 많은 수의 컨테이너들을 선박으로부터 선적 혹은 하역할 수 있느냐에 있으며, 이러한 컨테이너 처리능력의 향상을 위해서는 기존의 낙후된 항만시스템을 새로운 형태의 경쟁력 있는 시스템, 즉 자동화된 터미널로 바꾸어야 할 필요가 있다. 그리고 이미 일부 항만에서는 자동화된 ATC(automated transfer crane)와 AGV (automated guided vehicle)를 활용한 지능형 자동화 항만을 운영하고 있다. 그러나 자동화에 앞선 문제점은 아직까지 기존의 설비들이 나름대로의 기능을 발휘하고 있을 뿐만 아니라, 기존의 시스템을 자동화된 시스템으로 교체하기 위해서는 많은 경비와 시간이 소요되는 문제점이 있다. 이러한 문제의 해결방안으로 기존 설비의 재활용과 효과적인 교체를 통한 경비절감과 손쉬운 유지, 보수를 위해서 여러 가지 방법들이 간구될 수 있다. 본 논문에서는 크레인의 자동화 과정에서 기존의 운전자를 대체할 제어시스템으로 PC기반의 개방형시스템을 도입하여 전술한 문제들도 해결하고, 더욱 효과적인 자동화설비 구축이 가능함을 보이고자 한다. 그리고 자동화된 크레인은 주어진 태스크에 의해 컨테이너를 자동으로 선적 및 하역작업을 수행하기 때문에 자율주행하는 머니퓰레이터(로봇)와 같은 맥락으로 보고, 로봇의 제어구조를 크레인에 접목하여 작업의 효율성과 안정성을 높이고자 한다.

개방형시스템이라고 하는 것은 각각의 제조회사 마다 자사 전용시스템을 사용하는 폐쇄구조(closed architecture)의 시스템들을 하나의 표준환경으로 통합운영하여 상호 호환성을 가지도록 하는 시스템을 말하는 것으로서, single-vendor에서 multi-vendor로의 전환을 의미한다. 개방형시스템 특징들은 통합성(integration), 이식성(portablity), 경제성(economically), 확

*정회원, hongkt@pusan.ac.kr 051)510-1481

**정회원, kshong@pusan.ac.kr 051)510-2454

장성(scaleablity)으로 표현할 수 있다 (Hong et al, 2001; Mo et al., 1996; Sperling & Lulz, 1995; Wright, 1995).

개방형시스템의 연구형태로는 서로 다른 장비들을 상호호환성 있는 구조로 표준화하여 기존의 장비들을 큰 변화 없이 재구현이 가능하도록 하는 개방형 FA시스템에 관한 연구가 선진국들을 중심으로 시작되었다 (Hong et al, 2001; Nilson, 1980; OSACA, 1996; OSEC, 2000).

통합된 로봇시스템을 구현하는데 있어 잘 정의된 제어구조가 필수적이듯이, 크레인의 자동화에 있어서도 전체시스템의 제어구조가 자율운행의 성능을 좌우할 수 있다. 이동형 시스템의 경우 원하는 기능을 얻기 위해서는 수많은 하드웨어와 소프트웨어 모듈을 필요로 하는데, 이들을 통일성 있게 구현할 수 있게 하는 것이 제어구조이기 때문이다. 따라서 효율적인 제어구조를 설계하기 위한 연구가 지금까지 활발히 진행되어 왔다 (Gat, 1998). 90년대 초반에 제시된 것이 deliberative계층과 reactive계층을 동시에 가진 혼합구조이다 (Arkin, 1995). 이 접근방식은 거의 동시에 여러 연구자로부터 동시에 발표된 개념으로 연구자에 따라 조금식 차이가 있긴 하지만 근래들어 제시된 방법의 대다수는 이 패러다임에 수렴하는 경향을 보이고 있다.

따라서, 본 눈몬에서는 기존 크레인의 자동화 및 자동화 크레인의 새로운 설계에 있어서 크레인시스템을 PC와 PLC를 혼합한 하이브리드 구조를 사용하여 안정성을 보장하는 분산적 구조로의 설계를 제안한다. 그리고 자동화된 크레인을 자율운행하는 로봇으로 보고 제어구조를 로봇에서 효율적으로 사용되는 deliberative/reactive 하이브리드 구조로 구성한 후 개방형제어구조를 도입하여 하이브리드 제어구조를 구성하고 있는 모듈들을 객체지향형으로 모델링한다.

본 논문의 구성은 다음과 같다. 2절에서는 기존 크레인시스템의 구조와 문제점을 살펴보고 자동화크레인의 구성요소와 하드웨어 운용제어구조를 다룬다. 3절에서는 자동화크레인의 개방형 설계를 위해 OSACA에 기반한 표준기준모델을 제시하고, 4절에서는 자동화크레인을 위한 deliberative/reactive 하이브리드 제어구조를 제시하며 이를 표준기준모델을 통해 객체지향방법으로 모델링한다. 마지막으로 5절에서는 연구결과를 요약정리한다.

2. 자동화 크레인

전세계의 해상교역량의 증가와 함께 화물의 물동량이 급속히 증가하고 있으며 각 항구마다 화물을 빠르고 신속하게 처리할 수 있는 고속고성능의 크레인의 수요 또한 급격히 늘어나고 있는 추세이다. 그러나 기존의 크레인은 전문인력에 의해 운영되고 있긴 하지만 전문인력의 수가 부족할 뿐만 아니라 사람에 의해 운행되므로 정밀도와 정확도가 떨어져 작업능률이 극대화 되었다고 볼 수 없다. 또한 전문인력의 작업시간마저 무한정 늘릴 수 없다. 따라서 크레인의 자동화를 통하여 항구마다 최소의 크레인으로 최대의 작업생산성을 도모하고 운전자의 부족에 대처하기 위해서 항만 하역시설의 자동화는 필수적으로 요구되고 있다.

2.1 수동크레인의 구조와 문제점

기존의 크레인시스템은 운전자에 의해 컨테이너를 선적 및 하역하는 구조로서 사람이 크레인의 두뇌역할을 수행한다. 그러나 화물의 물동량이 급속히 증가하고 물류시스템이 지능화되는 현실에서 사람이 운행하는 수동크레인으로 항만의 경쟁력을 확보한다는 것은 불가능한 일이다. 그림 1에서 보는 바와 같이 컨테이너크레인은 운전자가 control stick을 조작하여 control signal을 발생시키며, PLC에 전달된 신호는 모터드라이버를 제어하여 크레인을 움직인다. 따라서, 주어진 작업을 수행하기 위해서는 운전자의 조작이 필수적이며 곧 운전자의 능력이 크레인의 하역능력으로 대변된다.

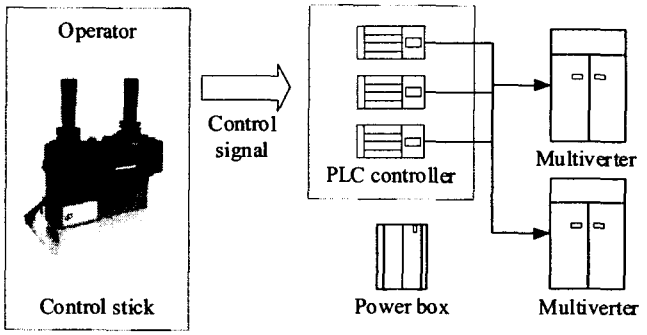

Fig. 1. A hardware platform of a manual operation crane.

2.2 자동화 크레인

자동화크레인은 운전자의 조작없이 주어진 선적 및 하역작업을 AGV와 연계하여 자율적으로 수행하는 것이라 할 수 있다. 따라서 기존의 수동크레인의 개조 및 새로운 크레인의 설계에 있어서 운전자를 대체할 수 있는 시스템을 구성하는 것이 최대의 관건이다. 또한 이러한 시스템은 개발 및 유지보수 그리고 운행단계에서 높은 유연성과 지능성을 보유하여야 하며 작업의 특성상 최대의 안정성을 확보하여야 한다. 따라서 크레인의 자동화 시스템은 높은 레벨의 자동화와 개방화가 요구되고 시스템 변화에 쉽게 사용자 설정이 가능해야 한다.

본 논문에서는 자동화하기 위해 운전자를 대신하여 동일한 설비투자에 대해서 효율을 극대화 할 수 있는 PC기반의 개방형시스템으로 대체하는 방법을 제안한다. 이는 기존의 제어기가 통합 및 확장 시에 많은 비용과 노력의 재투자가 필요하였지만 제어기의 구조를 개방화 함으로써 사용자의 적은 노력으로도 다양한 목적으로 제어기의 변경이 용이하도록 설계하고, 하드웨어와 소프트웨어의 구조적 모델을 제시한다.

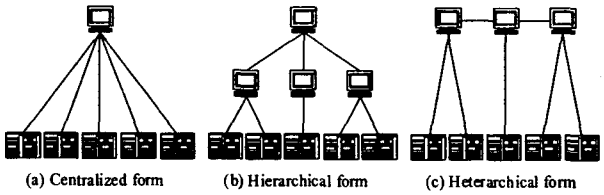

Fig. 2. A control system architecture

2.3 자동화시스템의 운용제어

자동화시스템의 운용제어는 두 가지로 정의할 수 있다. 첫째는 기계나 설비 수준에서 이들이 목적에 알맞은 동작을 하도록 조절하는 것이다. 둘째는 기계나 설비가 모여 시스템을 이루었을 때 시스템의 시작 및 종료, 필요한 작업을 수행하기 위한 명령생성, 기계로부터의 피드백 정보분석, 고장회복 등의 기능을 말한다 (Groove, 1980). 일반적으로 운용제어구조의 유형은 중앙집중구조, 계층구조, 분산구조로 나눌 수 있다. 그림 2는 각각의 운용제어구조를 보여주고 있다.

중앙집중 운용제어구조는 한대의 중앙컴퓨터가 현장에서 이루어지는 모든 생산 활동을 계획하고, 관련된 정보들의 처리 및 유지를 담당한다. 이 구조의 장점은 시스템에 필요한 모든 관련정보들을 하나의 제어기에서 처리하기 때문에 전체적인 최적화를 가능하게 하며 다른 방식에 비해 제어기 비용을 줄일 수 있고 관리가 용이하다. 그러나 시스템의 규모가 커지는 겨우 중앙컴퓨터에 부하가 많이 걸려 상황에 실시간으로 대처하는 능력이 떨어지게 되며, 중앙 컴퓨터에 이상이 생겼을 경우 전체시스템에 문제가 발생하는 단점이 있다.

계층적 운용제어는 구조가 계층별로 구성되어 있기 때문에 소프트웨어를 개발하는데 있어 모듈화가 가능하며 각각의 계층별로 역할이 분담되어 있기 때문에 상황에 대처하는 반응시간이 빠르다. 반면에 단점은 하위단계는 상위단계에 의존하게 되므로 어떤 부분에 이상이 생기면 그 하위단계의 제어기들도 기능이 마비되며, 시스템구조를 초기 디자인 단계에서 확정, 설치하므로 미래의 예측하지 못한 변화에 대처해서 수정, 확장하기 어렵다는데 있다.

이러한 계층적 운용제어의 단점을 극복하기 위해 통신망과 분산 컴퓨팅 분야의 기술 발전을 배경으로 한 분산적 운용제어구조가 제안되고 있다. 이 방식은 시스템 내에 계층적 구조가 존재하지 않는 대신, 시스템을 구성하는 구성원들 사이의 협력을 통해 원하는 목적을 달성하다. 전체적인 정보를 최소화시키고 대신에 지역 데이터베이스들을 활용하며, 제어기 간의 통신을 통한 협상 하에 의사결정이 이루어지는 운용제어 방식이다. 수평적 운용제어구조는 제어기들이 독립적으로 작업을 수행하기 때문에 플러그 앤 플레이 개념을 도입할 수 있어 시스템의 확장이 용이하며 특정 제어기의 이상발생이 전체 시스템에 미치는 영향을 최소화시킬 수 있다. 그리고 실시간 운용제어가 가능하며 오류상황 시에도 상당 수준의 동작이 가능하게 해 주는 fault tolerance를 갖는다. 본 논문에서는 크레인의 자동화를 위해 분산적 운용제어구조를 적용한다.

3. 자동화 크레인 표준기준모델

본 절에서는 기존의 개방형시스템을 바탕으로 해서 제안된 표준기준모델을 살펴본다. 이러한 모델은 크레인시스템이 객체지향형 개방구조를 가지도록 구성하기 위한 기준이 된다. 그림 3은 크레인제어시스템을 개방화하기 위해서 본 논문에서 사용하고 있는 표준기준모델을 보이고 있다. 이러한 모델은 하드웨어플랫폼, 운영시스템모듈, 응용소프트웨어모듈 등으로 구성되어 있다.

3.1 하드웨어 플랫폼

일반적으로 하드웨어 플랫폼은 계산을 담당하는 프로세서, 다른 외부장치로 입출력을 담당하는 I/O장치, 데이터 저장을 담당하는 메모리, 그리고 사용자 편의를 제공하는 주변장치들로 이루어져 있다. 이러한 하드웨어 플랫폼의 구성요소들을 개방화하기 위해서는 각 구성요소들이 표준화된 제원을 따르는 구조로 구성되어야 한다. 그러나, 오늘날 하드웨어가 빠른 속도로 발전하기 때문에 하드웨어 자체에 대한 제원을 표준화하기보다는 상황에 따라 가장 적합하고 가격이 저렴한 하드웨어를 어느 때고 쉽게 사용이 가능하게 하기 위해서는 하드웨어 드라이버가 개방화된 구조를 가져야 한다. 이러한 드라이버의 개방화를 위해서 최근 PC가 하드웨어플랫폼으로 도입되고 있다. 이와 같이 PC를 하드웨어플랫폼으로 도입하면 다음과 같은 장점을 가진다. PC는 하드웨어적인 측면에서 다양한 주변장치에 대한 적응성이 뛰어나기 때문에 최근에 개발된 하드웨어 기술들을 금방 도입할 수 있으며, 또한 하드웨어 드라이브 공급자 입장에서 쉽게 접할 수 있는 범용성을 갖추고 있다. 소프트웨어적인 측면에서는 구동프로그램을 모듈화하고 수정이나 확장성을 가지도록 설계하는 것이 용이하며, 빠르게 발전하는 소프트웨어 기술들을 쉽게 채용할 수 있다. 또한 PC를 하드웨어 플랫폼으로 사용하면 제어시스템의 가격저하와 고장시 교체가 용이한 부수적인 효과가 기대된다

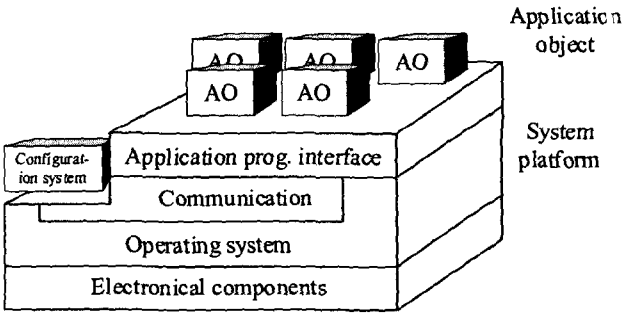

Fig. 3. A proposed standard reference model for the crane control system

3.2 운영시스템 모듈

개방형 제어시스템에서 운영시스템 모듈은 하드웨어, 응용소프트웨어, API (Application Programming Interface)등에 대한 유용성(availability)을 제공해 주어야 한다. 여기서 유용성은 제3자가 개발한 주변장치 및 소프트웨어에 대한 활용성을 나타낸다. 하나의 프로세서로 운영되는 PC를 개방형 제어시스템의 플랫폼으로 사용한다면 반드시 운영시스템 모듈이 실시간 운영시스템(*Real Time Operating System: RTOS*)으로 구성되어야 한다. 이러한 RTOS와 Windows/DOS의 중요한 차이는 CPU시간의 관리방법, 메모리의 할당방법, 인터럽트에 대한 응답과 기계장치를 구동하는 처리과정에서 자원공유방법 등에 달려 있다.

3.3 응용소프트웨어 모듈

개방형 제어시스템에서 응용소프트웨어 모듈은 필요에 따라 기능의 추가 및 변경이 허용되고, 각 구성요소간에 정보교환이 가능한 객체지향적 특성을 만족해야 한다. 이와 같은 특성을 만족하기 위해서 응용소프트웨어 모듈은 관리, API, 네트워크,

제어객체 등으로 구성된다.

(1) 관리(Management): 개방형 제어시스템의 가장 중요한 특징은 환경재설정 (reconfiguration)의기능이다. 관리는 이러한 개방형 제어기의 환경재설정의 역할을 수행한다. 일반적인 개방형 제어시스템의 환경 재설정과정은 다음과 같은 과정으로 이루어진다. 먼저 사용자는 off-line에서 실제 제어시스템의 기능 및 성능명세를 분석한다. 그리고 개방형 표준기준모델에 입력할 변수값을 환경설정편집기에 입력하여 환경설정주문(configuration order)을 작성한다. 그리고 환경설정주문에 포함된 변수값을 각종 제조장비의 구성요소에 대해 체계적으로 분류하여 구성한 데이터베이스로부터 가져와서 개방형 제어시스템의 환경을 재설정한다. 이러한 환경재설정 기능은 제조환경의 변화에 대응하기 위한 개방형 제어시스템의 필수요소가 된다.

(2) API: 제조환경은 생산제품이나 생산공정의 변화에 따라 제조장비의 재구성이나 추가적인 설치를 요구하게 된다. 이러한 제조환경의 변화에 따라 제어시스템의 구성요소도 재조정이 요구된다. 이와 같은 제어시스템의 유연성이나 확장성에 대한 문제는 개방형 제어시스템을 구성하는데 많은 어려움을 준다. 이러한 문제점에 대한 대처방안으로 하드웨어나 소프트웨어 측면에서 기존의 하드웨어나 소프트웨어와 공통된 인터페이스를 형성할 수 있는 범용의 API 모듈들을 시스템기반에 포함시킨다. 이러한 API모듈은 제어시스템의 모든 구성요소가 쉽게 재사용 또는 추가될 수 있도록 한다.

(3) 네트워크: 네트워크는 객체들의 현재상태의 값을 읽어 들이거나 필요한 동작에 대한 적절한 값을 보낼 때 데이터버스의 역할을 한다. 네트워크는 개념적으로 두개의 부분으로 나뉘어 데이터를 교환한다. 즉, 기능 수행을 위한 명령을 하달하는 클라이언트(client)와 클라이언트의 명령에 따라 요구된 기능을 수행하는 서버(server)이다 (Sperling & Lulz, 1995).

(4) 제어객체(Control Object): 제어객체는 제조장치를 실제로 제어하는데 사용되는 구성요소를 소프트웨어 측면에서 캡슐화하여 객체로 나타낸 것이다. 이와 같은 제어객체는 사용자가 객체의 속성을 재설정함으로서 새로운 생산환경의 제어시스템 구축이 가능하고, 필요에 따라서 추가 및 삭제가 가능해야 한다. 또한, 제어객체는 개방형 제어시스템의 하드웨어에 독립적이어야 한다.

4. 자동화 크레인의 객체지향형 모델

본 절에서는 앞에서 분석된 항만 컨테이너크레인의 구조를 이용하여 자동화 크레인의 제어구조를 설계한다. 그리고 설계된 제어구조를 제시된 표준기준모델과 같은 개방형구조를 가질 수 있도록 새로운 객체를 정의하고, 정의된 객체들을 상용의 하드웨어와 소프트웨어로 구현한다. 이와 같이 구현된 객체들을 PC상에서 통합하고, 객체의 속성들을 재설정할 수 있는 소프트웨어를 개발함으로서 PC기반의 개방형시스템을 완성한다.

4.1 하이브리드 제어구조

본 논문에서는 자동화크레인의 자율운행을 위해 하이브리드 제어구조를 도입한다. 그림 4는 자동화 크레인을 위한 하이브리드 제어구조를 나타낸다. 지능형 자동화 항만에서 사용되는 크레인은 중앙 통제실로부터 작업데이터베이스를 전달받아 다른 하역장비들과 연계하여 자율적으로 하역작업을 수행하여야 한다. 따라서 분산처리 개념을 도입하여 각 기능을 schema에 분산함으로써 병렬처리가 완벽하게 지원되는 reactive control 기반의 하이브리드 크레인 제어구조를 제안한다.

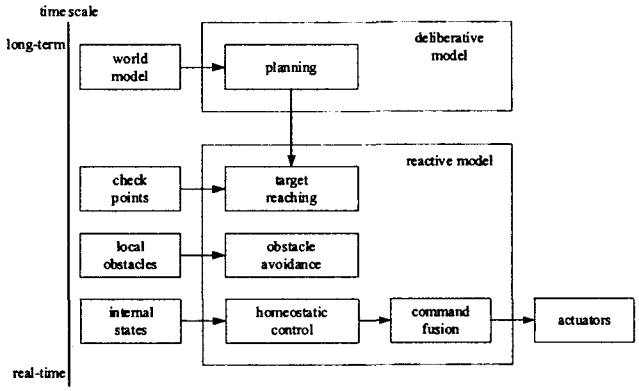

Fig. 4. A hybrid architecture for integrated task planning and control

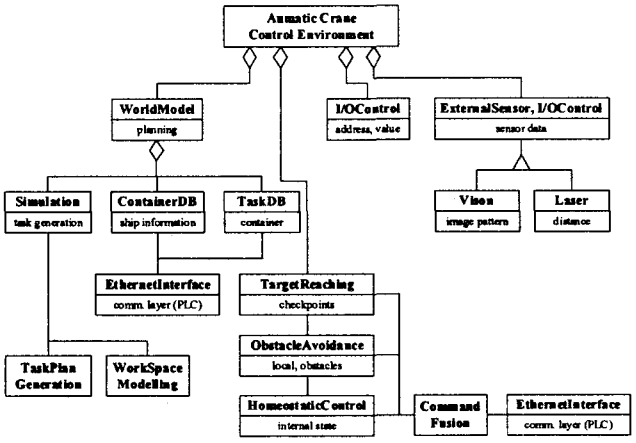

Fig. 5Class structure of the application software module

(1) Deliberative 모듈

크레인 제어시스템에서의 deliberative 모듈은 최상위 레벨의 제어기로서 크레인 자율주행전반을 담당한다. 최상위 레벨에서 deliberative planning module은 cell decomposition method의 변화를 사용하여 컨테이너의 장착지점에서 탈착지점까지의 장애물과 경로를 체크하고 최단거리의 경로를 생성한다. Planning module은 정확한 이동경로를 생성하고 작동기에 의해 실행될 행동들을 순차적으로 구성한다.

(2) Reactive 모듈

크레인 시스템에서 reactive 블록은 세 단계의 레벨로 구성된다. 첫 번째 레벨의 target reaching module은 하역작업을 위한 check point간의 최적이동경로를 구성한다. 하위단계의 두번째 레벨에서는 크레인, 컨테이너, 선박구조물 그리고 예측하지 못하거나 움직이는 장애물의 위치를 센싱하고 컨테이너가 장애물을 피해서 운행하도록 추가적인 제어명령을 만들어낸다. 가장 낮은 레벨의 homeostatic 제어모듈은 크레인의 내부상태를 센싱하여 환경변화의 보상에 따른 좌표화된 결과를

만들어 내부적 안정을 유지한다.

크레인의 동작제어를 위한 reactive 모델은 세 개의 하위레벨 모듈들로 구성된다. 그리고 command fusion 모듈은 reactive 요소로 부터의 제어입력을 받아 최종제어명령을 생성하여 작동기를 구동한다.

하이브리드 구조에서 모든 모듈들은 다른 속도비율로 비동기적 동작을 수행한다. Planning 모듈은 작업의 복잡 정도에 따라 몇 초에서 몇 분의 시간간격을 가지고 실행되며, obstacle avoidance 모듈이 대략 100ms의 간격으로 작동하는 동안, target reaching 모듈은 servo tick 사이에서 약 1초 간격으로 동작한다. Homeostatic 제어모듈은 1 msec 간격으로 작동한다. 그리고 command fusion 모듈은 제어명령들이 생성될 때 마다 작동한다.

4.2 응용소프트웨어 모듈의 구성

본 절에서는 컨테이너크레인의 하이브리드 제어구조를 개방형시스템으로 구현하기 위해서 모듈의 구성요소들을 각각 객체로서 정의하고 그 특성과 행위를 부여하는 객체지향적인 모델링을 수행한다 (Cost-Maniere & Simon, 1993). 그림 5에서는 객체지향 접근법의 OMT (Object Modeling Technique)를 사용하여 각 클래스 및 클래스들 사이의 논리적 관계가 정의되어 있다 (Rumbaugh et al., 1991). 이러한 객체지향적 모델링에서 모델링된 객체들에 대한 상호작용 및 역할과 구현방법 등을 상술하면 다음과 같다.

WorldModel 객체는 컨테이너크레인이 선적 및 하역작업을 수행할 수 있도록 작업 계획을 생성하고 작업계획에 따른 일련의 동작들을 정의하며, 이러한 task 생성을 위해서 ContainerDB객체, TaskDB객체, PathGeneration객체 그리고 Simulation객체와 상호작용한다. 여기서 ContainerDB객체는 EthernetInterface객체의 CommunicationLayer로부터 전달된 정보를 바탕으로 새로 접안한 컨테이너선에 관련된 정보를 재구성한다. 이때 재구성되는 정보는 컨테이너선의 제원과 화물의 선적상태 그리고 하역작업내용에 관한 것들이다. 컨테이너선박의 컨테이너 하역작업에는 여러기의 크레인이 동시에 작업을 수행하게 된다. 그리고 자동화된 크레인은 각자의 작업할당량을 가지고 자율적으로 하역작업을 수행한다. 따라서 새로 접안한 선박의 정보와 작업내용을 중앙통제실로부터 전달 받을 필요가 있다. EthernetInterface객체는 이러한 정보들을 중앙통제실로부터 무선네트웍을 통해 전달받아 ContainerDB객체와 TaskDB객체로 전달하는 역할을 수행한다. TaskDB객체는 EthernetInteface객체로부터 전달받은 작업내역리스트를 재구성하여 WorldModel객체와 Simulation객체로 넘겨주는 역할을 한다. PathGeneration객체는 WorkSpaceModel객체로부터 받은 컨테이너선박의 화물선적상태와 WorldModel객체로부터 받은 하역작업을 이용하여 스프레더의 3차원 이동궤적을 구성하여 WorldModel객체로 전달한다. 이러한 각 객체들의특징 및 역할들은 객체지향형 언어인 C++을 사용하여 각각 클래스들로 작성된다. 이와 같이 작성된 클래스들은 데이터 멤버(data member)들과 멤버함수(member function)들로 구성되어 있다. 데이터 멤버는 각 객체들과 관련된 데이터를 저장하는 역할을 수행하고 멤버함수들은 클래스들이 처리해야할 다양한 연산들을 수행한다. 여기서 멤버함수들은 통상적으로 구현자 함수(implementor function), 보조함수(helping function), 접근함수(access function)들로 구성된다.

Simulation객체는 WorkSpaceModel객체, ContainerDB객체, TaskDB객체로부터 작업대상인 컨테이너선박의 정보와 하역작업정보 그리고 주변환경조건을 전달받는다. 그리고 실제 크레인이 컨테이너 하역작업을 시행하기전에 제한조건에 근거하여 TaskDB의 작업내용을 최소의 시간에 효율적으로 수행하도록 시뮬레이션을 통해 재구성한다. Simulation객체가 WorkSpace Model로부터 전달받는 주변환경조건에는 작업대상선박의 위치와 동일선박에 대해 작업중인 다른 크레인에 대한 정보 그리고 육측의 컨테이너 운반차량의 이동경로 등이다. Container DB객체는 컨테이너선박에 적재된 컨테이너정보를 가지고 있어 Simulation객체가 효율적으로 하역작업을 구성할 수 있는 정보를 제공하며 또한 적재된 윤곽을 추출할 수 있는 정보를 제공하여 충돌방지에 효율적으로 활용될 수 있다. TaskDB객체는 컨테이너크레인이 수행하여야 할 하역작업내역을 데이터베이스로 구성하여 그 정보를 Simulation객체에 넘겨주어서 Simulation객체가 최적의 작업순서를 재구성하도록 한다. 컨테이너 크레인이 수행하여야 할 작업은 크게 두 가지로 분류되는데 컨테이너차량이 운반해오는 육측의 컨테이너를 컨테이너선박에 적재하는 선적작업과 컨테이너선박에서 육측의 컨테이너차량으로 컨테이너를 내리는 하역작업이 있다.

Reactive모델에 속하는 객체중 최상위 레벨의 Target Reaching객체는 deliberative모델의 구성요소인 WorldModel객체로부터 하나의 컨테이너에 대한 선적 또는 하역의 작업명령을 받아 스프레더의 이동경로간에check point의 최적 이동경로를 구성한다. ObstacleAvoidance객체는 크레인, 컨테이너, 선박구조물 그리고 예측하지 못하거나 움직이는 장애물의 위치를 센싱하고 컨테이너가 장애물을 피해서 운행하도록 추가적인 제어명령을 만들어낸다. HomeostaticControl객체는 내부상태를 센싱하여 환경변화의 보상에 따른 좌표화된 결과를 만들어 내부적 안정을 유지한다.

CommandFusion객체는 TargetReaching객체와, Obstacle Avoidance객체 그리고 HomeostaticControl객체로부터 각각 명령을 받아 컨테이너크레인 제어시스템에 설치된 각 구동기에 관한 환경 재설정작업을 수행하며, EthernetInteface객체를 통해 해당 구동기의 제어시스템으로 전송한다. 컨테이너크레인의 구동기로는 크레인의 이동, 트롤리의 이동, 스프레더의 상하이동 그리고 outreach를 들어올리기 위한 것들이 있다.

I/OControl객체는 입출력 인터페이스를 통해 보조장비(사용자 인터페이스, 모터 드라이버 등)들로부터 나온 신호를 입력포트를 통해 받고, 보조장비에 출력포트를 통해 신호를 보내는 작업을 수행한다. 이러한 작업은 인터페이스에 정해진 각 포트를 통해 정의되고, 하드웨어에 관련된 복잡한 상호작용의 세부관계는 캡슐화된다. 그러나, I/OControl객체가 NT상에서 각 입/출력 포터를 제어하기 위해서는 디바이스 드라이브가 구현되어야 한다.

ExternalSensorControl객체는 컨테이너크레인 제어시스템의 외부환경에 대한 정보를 외부 센서로부터 받아들여 필요한 필터연산을 수행하여 구체적인 정보를 생성하고, 이를 deliberative모델과 reactive모델이 인식할 수 있는 데이터형식

으로 전환한다. 구체적인 예로서는 비젼센서, 레이져센서 등을 들 수 있다.

4.3 PC상에서 각 모듈의 통합과 객체 상호 간의 정보교환

자동화 크레인시스템은 기존의 PLC기반의 수동시스템에서 운전자를 대신할 PC기반의 제어시스템을 추가함으로서 구현할 수 있다. 자동화된 크레인제어시스템은 PC를 기반으로 하는 하드웨어 플랫폼과 운영체제 모듈을 구축한 다음, 구현된 응용객체와 CODE 시스템, 기타 하드웨어장치를 PC에서 통합함으로서 구현된다. 또한 응용프로그램 모듈내에서 각 객체사이의 정보를 읽어들이거나, 전달하는 역할은 C++에서 제공되는 TCP/IP 프로토콜을 사용하여 windows NT 운영체제 내에서 내부통신의 형태로 구현된다. 이와 같이 각 객체간의 정보교환을 네트워크를 사용하여 구현하는 목적은 각 객체의 추가/변경시에 개방성 및 독립성을 보장하기 위해서이다. 또한 제안된 컨테이너크레인 제어시스템은 TCP/IP를 통하여 각 객체들의 속성을 재설정하는 것이 가능하도록 구성되었다. 이와 같이 구성된 각 객체들은 컨테이너크레인 제어시스템에서 정의된 프로세서 작업을 수행하고, 제어시스템의 구조를 변경 또는 추가할 때 유연성 및 확장성을 제공한다. 예를 들면, 자동화 성능을 향상시키기 위해 새로운 알고리듬과 모듈을 추가할 경우 특정 모듈의 수정과 새로운 모듈의 추가로서 제안된 제어시스템은 빠르게 업그레이드 된 새로운 시스템을 재구축하는 것이 가능하다

5. 결 론

본 논문에서는 기존의 컨테이너크레인을 자동화하기 위해 운전자대신 PC를 사용하여 크레인제어시스템의 제어기능들을 하나의 환경으로 통합하며 크레인을 자율동작하는 로봇으로 간주하여 deliberative/reactive 하이브리드 제어구조를 적용한 컨테이너크레인 제어시스템을 제시하였다. 본 논문에서 제시한 컨테이너크레인의 제어시스템은 세 가지 측면에서 요약할 수 있다. 우선 크레인시스템의 하드웨어 측면에서 전체 시스템의 하드웨어 운용제어구조를 분산형으로 제안하여 기존의 PLC시스템을 그대로 활용하면서 자동화 크레인의 두뇌인 운전자를 대체할 수 있는 PC기반의 제어시스템 만을 추가하였다. 다음으로, 소프트웨어 측면에서 살펴보면 PC기반의 제어시스템에서 운영될 소프트웨어는 운전자의 지능을 대체하며 크레인이 자율적으로 하역작업을 수행할 수 있게 한다. 따라서 전체 소프트웨어의 구조를 로봇시스템에서 활용되고 있는 deliberative /reactive 하이브리드구조로 제안하였다. 마지막으로 제안한 하이브리드 제어구조를 OSACA에 기반한 표준기준모델을 바탕으로 객체지향형방법으로 모델링하여 개방구조의 제어기를 구축하였다.

후 기

본 연구는 과학기술부의 국가지정연구실사업(과제번호: M1-0302-00-0039-03-J00-00-023-10)의 지원에 의하여 수행되었으며, 이에 관계자 여러분께 감사 드립니다.

참 고 문 헌

[1] R. C. Arkin (1995), "Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation," Robotics and Autonomous System, vol. 6, no. 6, pp. 24-31.

[2] E. Coste-Maniere and D. Simon (1992), "Reactive Objects in a Task Level Open Controller," IEEE Int. Conf. on Robotics and Automation, pp. 2732-2737, Nice, France.

[3] E. Gat (1998), Three-Layer Architectures in AI and Mobile Robots, AAAI Press.

[4] M. P. Groove (1980), Automation, Production Systems, and Computer- Integrated Manufacturing, Prentice-hall.

[5] K. S. Hong, K. H. Choi, J. G. Kim and S. Lee (2001), "A PC-Based Robot Control System: PC-ORC," Robotics and Computer Integrated Manufacturing, vol.17, no. 4, pp.355-365.

[6] J. P. T. Mo, Y. Wang, and C. K. Tang (1996), "The Use of the Virtual Manufacturing Device in the Manufacturing Message Specification Protocol for Robot Task Control," Computers in Industry, vol. 28, pp. 123-136.

[7] N. J. Nilson (1980), Principles of Artificial Intelligence, Palo Alto: Tioga.

[8] OMAC API Work Group (1998), OMAC API SET-Working Document, OMAC API Work Group, Ver. 0.18.

[9] OSACA (1996), Open System Architecture for Controls within Automation System, ESPRIT Project 6379/9115.

[10] OSEC (2000), Open System Environment for Controller, http://www.sml.co.jp.

[11] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991), Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ.

[12] W. Sperling and P. Lulz (1995), "Enabling Open Control Systems: An Introduction to the OSACA System Platform," ESPRIT III Project: Stuttgart: FISW GmbH.

[13] P. K. Wright (1995), "Principles of Open-Architecture Manufacturing," Journal of Manufacturing Systems, vol. 14, no. 3, pp. 187-202.