*i*Maker: 시뮬레이션 기반의 제어 소프트웨어 디자인 도구

심민석, 유대승, 박성규, 김종환, 이명재
울산대학교 컴퓨터·정보통신공학부
{sms, yds, icoddy, bearknight, ymj}@mail.ulsan.ac.kr

*i*Maker: Simulation based Design Tool for Instrument Control Software

Minsuck Sim Daesung Yoo Sunghue Park Jonghwan Kim Myeongjae Yi
School of Computer Engineering & Information Technology, University of Ulsan

요 약

오늘날 장비 제어 기술은 장비 제어 시스템 개발의 비용을 효과적으로 줄이고 유연하고 효율적인 개방 구조로의 전환이 요구되고 있다. 그러나 대부분의 장비 제어 시스템들은 그들 자신의 구조를 기반으로 각기 다른 방법으로 구현이 되어 있어 시스템의 개발 및 통합에 많은 비용이 든다.

본 논문에서는 장비 제어의 표준으로 등장하는 OPC를 이용하여 장비 구성요소들 간의 복잡한 활동을 디자인하고 시뮬레이션 하는 디자인 도구(***i*Maker**)에 관하여 기술한다. 또한 XML 기술을 사용하여 OPC 기반의 장비의 특성과 행위를 모델링 하였다. ***i*Maker**는 장비의 행위를 미리 구성하고 시뮬레이션을 통하여 장비 개발의 질을 향상시키고 발생 가능한 문제를 미리 예측할 수 있으므로 장비 제어 시스템의 개발 및 유지보수에 대한 비용을 현저히 줄일 수 있다.

1. 서론

현대의 생산시스템은 제품에 대한 소비자들의 다양한 요구를 만족시키고, 생산기술의 변화와 컴퓨터 관련 기술의 급격한 발달에 빠르게 적응하기 위해서 유연성(flexibility), 통합성(integration) 및 동시성(concurrency)을 만족시키는 개방구조(open architecture)로의 전환이 요구되고 있다[1][2]. 또한 복수의 공정장비 또는 모듈과 로보트등의 유연한 물류장비가 밀접하게 결합되어 하나의 제조셀 또는 복합장비를 이루는 경향이 있는데[3], 이러한 생산시스템의 흐름에 XML (Extensible Markup Language)과 OPC(OLE for Process Control) 기술이 결합되어 유연하게 장비들간의 수직 및 수평 통합되는 큰 흐름을 형성하고 있다.

본 연구는 심[4]이 제시한 효율적인 제어 소프트웨어 생성 방법 중 OPC 기반의 장비 구성요소들 간의 복잡한 활동 및 흐름에 신속하게 대처하는 시뮬레이션 기반의 디자인 도구(***i*Maker**)에 대해 기술한다.

논문의 구성을 살펴보면 다음과 같다. 2장은 시뮬레이션 기반의 장비 제어에 연관된 관련 연구를 알아보고 3장에서는 시뮬레이션 기반의 제어 소프트웨어 생성 시스템을 구성하는 기술적인 배경에 대해서 알아본다. 4장에서는 시뮬레이션 기반의 복합장비 제어 시스템(***i*Maker**)을 디자인 및 구현한다. 5장에서는 제어 소프트웨어 생성 과정을 기술하고 6장에서는 결론과 향후 연구 과제에 대해서 논한다.

2. 관련연구

시뮬레이션 및 동작 관련 연구들은 FY2002 프로젝트[5]를 중심으로 2005년까지 제조시스템의 시뮬레이션, 시뮬레이션과 실제 장비와의 일치 테스트, 표준 인터페이스에 대한 주제로 진행하고 있으며 디자인 도구 기반의 시뮬레이션에 대해 다루고 있다. *Hui*[6]는 에너지 디자인과 시뮬레이션을 다루었으며 Douglas A. Bodner[7]은 구조적 접근 방법을 이용하여 생산 시스템에 대한 시뮬레이션을 모델링 하는 연구를 하였다.

이외에 OPC 기반의 장비 제어 관련 연구들은 OPC에 대한 일반적인 개념 소개에 초점이 맞춰 있으며[8][9], 특별히 Matthias Riedl [10]는 OPC 기술을 사용하여 다양한 장비의 드라이브 통합 구조를 제안하였다. 그러나 이러한 대부분 연구들은 주로 OPC 개념의 소개 정도에 머물고 있으며 장비들의 복합적인 행위를 표현하는 부분에 대해서는 다루고 있지 않았다. 특히 OPC 기반 핵심 기술인 OLE/COM의 장점(데이터의 동적 교환, 오토메이션, 커넥션포인터, 모니커,..)을 적용하고 응용한 연구는 전무한 편이다. 이외에 자바를 이용하여 사용자 인터페이스(UI)를 구성하고 JNI(Java Native Invocation) 기술을 사용하여 장비를 제어하는 방향의 연구분야도 있었다[11].

위와 같은 연구를 통하여 장비 제어 소프트웨어 생성에 대한 관련 연구 동향이 XML과 OPC 중심으로 빠르게 발전하고 있으며 이러한 기술이 중요한 요소가 되고 있음을 알 수 있다. 본 논문에서도 이와 같은 최근 기술 발전 추세를 감안하여 장비들의 특성을 XML과 OPC 기술을 기반으로 표현하여 이러한 기술들이 포함하고 있는 이점들을 수용하고 활용함으로써 유연하고 빠른 시스템 설계 및

본 연구는 한국과학재단 지정 울산대학교 네트워크기반 자동화연구센터 지원에 의해 이루어 졌습니다.

개발을 가능하게 하고자 한다.

3. 기술적인 배경

***i*Maker**는 이기종간의 장비 및 환경 사이의 통신 및 통합과 재사용성을 고려하여 컴포넌트 기반의 소프트웨어 개발 방법을 사용하여 디자인 및 구현하였다. 또한 장비 제어의 표준으로 떠오르고 있는 OPC(OLE for Process Control) 기술과 OPC 제어 장비의 특성을 표현하는데 XML 기술을 사용한다.

컴포넌트 기반의 소프트웨어 개발 기술은 다변화 및 급변화에 따른 소프트웨어 생산성 향상에 부응하기 위한 기술이며 컴포넌트 단위로 독립적인 개발이 가능하기 때문에 병행적, 단계적 소프트웨어 개발을 통한 개발 기간 단축 효과를 얻을 수 있다. 또한 컴포넌트의 수행은 컴포넌트 플랫폼 아키텍처상에서 이루어지므로 플랫폼 아키텍처가 제공하는 트랜잭션, 보안성, 지속성 등의 다양한 서비스를 이용할 수 있으며 컴포넌트 단위로 수행되므로 컴포넌트의 대체 및 유지보수가 수월한 장점을 가진다.

OPC(OLE for Process Control)는 마이크로소프트사의 WinSEM 그룹(Windows for Science, Engineering, and Manufacturing)에서 시작하여 OLE/COM 기술을 바탕으로 프로세스 데이터의 클라이언트 어플리케이션들과 서버(장비)들 사이의 인터페이스 방식을 규정한 것이다. OPC 서버의 장비 제어 구조를 살펴보면 그림 1과 같다. OPC 기반 장비의 제어 방식은 장비의 행위를 제어하는 프로토콜 부분과 프로토콜의 상위에 존재하면서 장비를 제어하는 드라이버 부분과 컴포넌트 부분이 있다. 컴포넌트 부분은 장비의 특성을 기준으로 여러 개의 그룹으로 분류하고 각각의 그룹은 장비의 행위와 연동이 되어있는 태그로 구성된다. OPC 서버 컴포넌트는 장비제어의 새로운 부분이 아니라 기존에 사용하는 장비드라이버를 마이크로소프트사의 COM(Common Object Model) 기술을 사용하여 한 단계 더 추상화시킨 것이다. 이런 컴포넌트를 사용하게 되면 컴포넌트가 가지는 여러 가지 장점을 가질 수 있고 사용자는 장비제어를 위해서 장비의 프로토콜이나 장비 API 학습에 따른 비용을 줄일 수 있다.

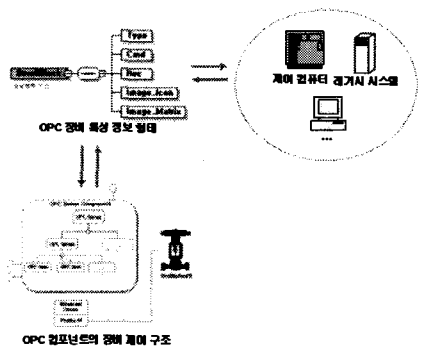

그림 1 OPC와 XML 관계

XML은 전자문서의 논리적 정보와 물리적 정보가 같이 표현되던 구조에서 문서의 논리적 구조(XML), 물리적 구조(DTD, XSD), 문서의 연결(Xlink, XPointer)을 분리하려는 요구와 시도에 의해 생성되었고 논리적인 구조의 표현 특성으로 여러 가지 분야에서 적용되어 사용하고 있다. 특히 제조 영역에서 XML은 장비의 특성을 기술하여 장비들 사이의 수평적인 통합 및 장비들과 레거시 시스템과의 수직적인 통합에 사용되어 핵심기술로 자리를 잡고 있다. NASA에서는 XML 기반의 장비 특성을 기술한 Instrument Markup Language (IML)[12]를 제안하였고 이와 관련하여 원격 장비 제어를 위한 Astronomical Instrument Markup Language (AIML)[13]을 정의하였다.

본 연구에서는 장비를 제어하는 OPC 컴포넌트의 장비 제어 구조(Block)를 기술하고 조합하는데 XML기술을 사용한다. OPC와 XML 기술은 분산환경의 이기종 장비 제어 시스템과 장비들간의 수직 및 수평 통합에 아주 많은 이점을 가진다.

4. *i*Maker 시스템

4.1 시스템 아키텍처

생산 시스템의 일반적인 구성은 구동 장비들이 연결되어 있는 필드 영역 , 장비를 제어하는 매니저 영역 그리고 어플리케이션 영역으로 분류할 수 있다.

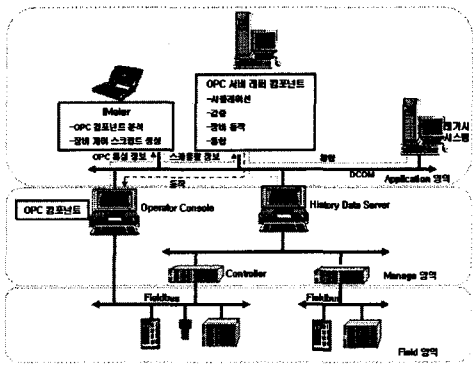

그림 2 시스템 아키텍처

***i*Maker** 시스템은 어플리케이션 영역 부분에 존재하며 원격에 존재하는 매니저 영역의 컴퓨터(Operator Console)로부터 장비의 특성정보를 추출하고 재구성하여 장비 제어 소프트웨어를 생성한다. 또한 분산 환경에서의 다중 장비의 트랜젝션과 레거시 시스템들과의 연동을 위하여 마이크로 소프트사의 컴포넌트 플랫폼 아키텍처(COM+)을 사용한다. ***i*Maker**는 매니저 영역의 컴퓨터에서 제어하는 장비들의 특성을 추출하고 장비의 엑션을 스케줄링 한다. 생성한 장비 스케줄 스크립트는 OPC 서버 랩퍼를 이용하여 시뮬레이션하며 스케줄 스크립트의 로직과 성능을 검증 할 수 있다. 또한 원격에 존재하는 장비의 제어를 위하여 DCOM 프로토콜을 이용하며, XML 기술을 사용하여 레거시 시스템과의 결합 구조(Connector)를 가진다.

4.2 *i*Maker 프레임웍 구조

***i*Maker** 프레임웍의 구조는 XML 기술을 사용하여 장비의 특성 정보(Small Block, Control Block, BigBlock)와 스케줄링 정보(Action Schedule Script)를 정의하는 부분, OPC 서버를 입력받아 장비의 특성을 추출하여 장비 스케줄링 스크립트를 생성하는 부분 그리고 스케줄링 스크립트 시뮬레이션 및 검증 작업 부분으로 분류한다.

장비 스케줄링 스크립트를 생성하는 부분은 크게 단순 블록 생성기(Small Block Maker), 복합 블록 생성기(Big Block Maker)와 장비 제어 스크립트 생성기(Action Schedule Maker)로 구성된다. 단순 블록 생성기는 OPC 서버로부터 장비의 특성 정보를 추출하는 모듈(Instrument Attribute Mining), 특성 정보를 탐색하고 에디팅을 통하여 블록으로 구성하는 모듈

본 연구는 한국과학재단 지정 울산대학교 네트워크기반 자동화연구센터 지원에 의해 이루어 졌습니다.

(Browsing/Editing), 블록이 가지는 제약사항을 생성하는 모듈(Constraints generating)로 구성한다. 복합 블록 생성기는 단순 블록 집합과 컨트롤 블록 집합을 이용하여 장비의 제어 스케줄링과 제약사항을 생성하는 모듈로 구성된다. 장비 제어 스크립트 생성기는 복합 블록과 컨트롤 블록을 이용하여 장비 스케줄 스크립트를 생성하는 모듈로 구성된다.

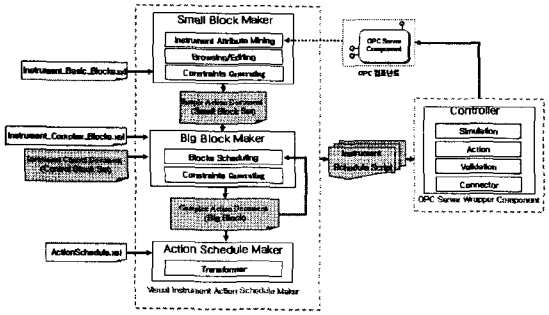

그림 3 *i*Maker 시스템 구조

시뮬레이션 및 검증 작업 부분은 시뮬레이션 모듈, 검증 모듈, 동작 모듈과 레거시 시스템과 결합하는 부분으로 구성된다.

4.3 장비의 제어 정보

장비 제어의 기본 단위는 블록(Block)이며 단순 블록, 복합 블록, 컨트롤 블록으로 분류된다.

단순 블록(Small Block)은 장비의 기본 동작 단위이며 OPC 컴포넌트에서 장비 특성 데이터(태그)를 추출하는데 이렇게 추출한 각각의 태그와 그래픽 정보(아이콘)가 더해져서 하나의 단순블록이 생성된다. 아래의 그림 4는 XML을 사용하여 기술한 단순블록의 형태이다.

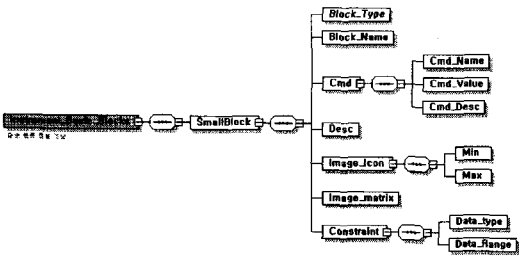

그림 4 단순 블록의 구조

컨트롤 블록(Control Block)은 블록의 흐름을 제어하는 기본 단위이며 시작과 종료에 관련된 기본 블록(Primitive Block), 동기화 컨트롤 블록(Sync Block), 제어 관련 블록(Control Block), 시간 관련 블록(Time Block)등 4가지 영역으로 분류하였다. 명령 흐름과 제어를 표기하는 연구는 Douglas[13]가 제안한 컨트롤 모델(Controller model)에서 약간의 언급이 있었으나 정확함 명세는 부족하며 이외의 연구 또한 미비한 편이어서 그림 5에 나타난 것처럼 UML의 엑티비티 다이어그램 요소들을 수렴하고 응용하였다.

복합 블록은 여러 장비의 행위들이나 단일 장비의 여러 동작들을 하나의 기본 단위로 정의하며 그림 6과 같은 구조를 가진다.

제어	표기법	설명
시작	●	행위의 시작
끝	◉	행위의 끝
분산	▬	여러 개의 들어 오는 전이와 하나의 나가는 전이
합침	▬	하나의 들어 오는 전이와 여러 개의 나가는 전이
판별	◇	X은 조건을 못하며 Y은 조건 만족, N은 조건 불만족
루프	◇	은 반복 카운터를 의미
지연	⧖	특정 시간 동안 기다림
기다림	⧖	이벤트나 특정 신호를 기다림

그림 5 제어 명령 집합

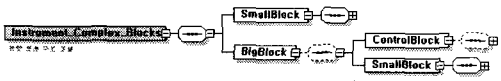

그림 6 복합 블록 구조

4.4 단순 블록 생성 마법사

단순 블록 생성 마법사를 사용하여 블록을 생성하는 사용자 인터페이스 모습(그림 7)이다. 단순 블록 생성 마법사 실행 시 로컬 네트웍 환경의 컴퓨터와 현재 연결되어 있는 OPC 서버 리스트의 목록들을 리스트 컴포넌트를 이용해서 보여준다. 단순 블록 생성을 원하는 OPC 서버를 선택하면 장비 특성 데이터(태그)를 추출하여 장비 특성 데이터 리스트 항목에 보여지며 추출한 각각의 태그와 그래픽 정보(아이콘)가 하나의 단순블록이 되고 태그 값에 따라 다양한 행동 패턴을 갖는다.

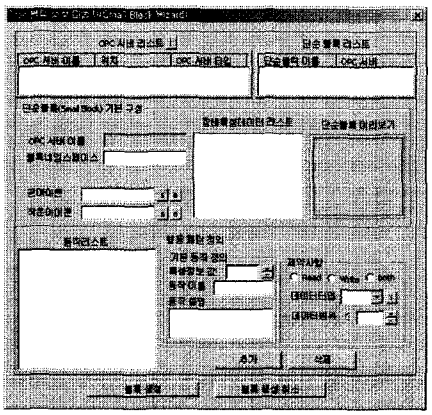

그림 7 단순 블록 생성 마법사

4.5 복합 블록 생성 마법사 & 장비 스케줄 스크립트

복합 블록 생성 마법사를 사용하여 블록을 생성하는 사용자 인터페이스 모습(그림 8)이다. 왼쪽의 툴바(1) 부분은 블록들의 리스트를 보여주는 부분이며 크게 단순 블록(1-1), 컨트롤 블록(1-2), 그리고 복합 블록(1-3)으로 구성된다. 중앙의 부분(2)은 복합블록을 디자인하는 화면이며 이렇게 생성한 블록은 재사용 가능하며 장비 행동 스크립트(Instrument Action Schedule Script) 형식으로 변환된다. 오른쪽의 툴바(3)는 블록의 특성과 행위를 설정하는 속성 창이다. 아래쪽의 툴바(4)는 생성 시 발생하는 메시지와 시뮬레이션 시 동작하는 결과를 모니터링 하는 화면이다.

본 연구는 한국과학재단 지정 울산대학교 네트워크기반 자동화연구센터 지원에 의해 이루어 졌습니다.

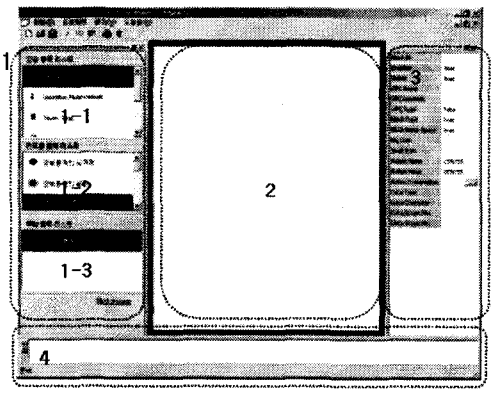

그림 8 복합 블록 디자인 및 모니터링 환경

생성한 복합 블록은 변환기를 통하여 장비 스케줄 스크립트를 생성하며 구조는 그림 9와 같은 구조를 가진다.

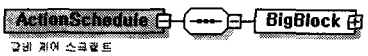

그림 9 장비 제어 스크립트 구조

4.6 OPC 서버 래퍼

OPC 서버 래퍼는 컴포넌트 기술 중 포함(Containment) 기법을 사용하여 OPC 컴포넌트 감싸고 있는 컴포넌트이며 OPC 컴포넌트가 제공하는 인터페이스 외에 스케줄을 입력받아 시뮬레이션하고 실제동작과의 검증 작업을 처리하는 IControl 인터페이스, 기존의 시스템 또는 컴포넌트와의 연결을 담당하는 IConnector 인터페이스 등 추가적인 인터페이스를 제공한다.

5. 제어 소프트웨어 생성 프로세스

제어 소프트웨어 생성 프로세스는 ***i*Maker**를 이용하여 제어를 원하는 장비의 특성 정보를 분석 및 추출하고 컨트롤 블록과 복합 블록을 사용하여 생성한 후보 장비 스케줄 스크립트는 로직 검증, 행위 검증 및 성능평가를 과정을 통하여 장비 행위에 가장 알맞은 스크립트를 결정한다.

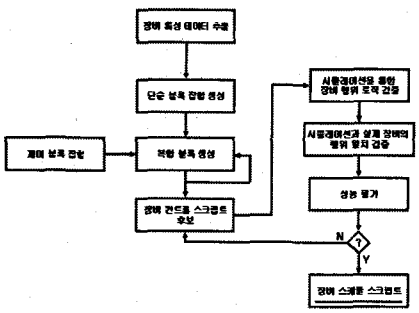

그림 10 제어 소프트웨어 생성 프로세스

본 연구를 통하여 생성한 스크립트 생성 기술과 시뮬레이션 기술은 가상 기계에 대한 검증과정을 통하여 실제 장비에 적용 시 장비들의 오류를 미리 예측할 수 있고, 통신 지연으로 인한 메시지 소실 및 제어 컴퓨터의 처리지연 등과 같은 예상치 못한 사건이나 소프트웨어 오류 발생들로 인한 장애 처리 부분의 핵심 기술로 자리 잡고 있다.

6. 결론

효율적인 제어 소프트웨어 생성 프레임웍은 OPC 서버 컴포넌트에서 장비의 특성 정보 추출하고 XML 기술을 사용하여 장비의 행동 정보를 표현 및 구성하는 구조를 가진다. 또한 장비의 행위를 시뮬레이션을 통하여 행위의 정확성과 성능을 검증하고 발생 가능한 문제를 미리 예측할 수 있으므로 장비 제어 소프트웨어 질을 향상시킬 수 있고 유지보수에 대한 비용을 현저히 줄일 수 있다.

본 논문에서 제안한 방법은 생산기술의 변화와 컴퓨터 관련 기술의 급격한 발달에 빠르게 적응하고 유연성(flexibility), 통합성(integration) 및 동시성(concurrency)을 만족시키는 개방구조(open architecture)로의 전환에 있어서 중요한 기술이 될 것이라 생각된다.

향후 연구과제로는 OPC 서버 레퍼의 모니터링 부분, 기존 시스템과 연동 부분, 실제 장비와 시뮬레이션 및 검증 부분 등의 연구와 장비들간의 수평 및 수직적인 통합에 관한 연구가 수행되어야 한다.

참고문헌

[1] W.Sperling and P.Lulz, "Enabling open control systems: An introduction to the OSACA system platform" , ESPRIT III Project: Stuttgart: FISW GmbH, 1995

[2] P.K. Wright, "Principles of open-architecture manufacturing" , Journal of Manufacturing System, vol. 14, no.3, pp. 187-202, 1995

[3] 이태억, "모델 기반의 장비 및 제조시스템 통합 기술" , LG생산기술, 2000

[4] 심민석, 유대승, 박성규, 기종환, 이명재, "OPC 기반의 유연하고 효율적인 제어 소프트웨어 생성에 대한 연구" , 한국 정보과학회 춘계학술발표대회 논문집, 2003

[5] FY2002 Projects, "Manufacturing Simulation & Visualization" , http://www.mel.nist.gov/proj/msv.htm

[6] Hui, S. C. M., 1998. Simulation based design tools for energy efficient buildings in Hong Kong, Hong Kong Papers in Design and Development, Vol. 1, 1998, pp. 40-46, Department of Architecture, University of Hong Kong.

[7] Douglas A. Bodner and L. F. McGinnis. "A Structured Approach to Simulation Modeling of Manufacturing Systems," Proceedings of the 2002 Industrial Engineering Research Conference, 2002.

[8] Matthias Riedl, Mario Thron, Thomas Hadlich, "DriveServer-significantly reduce in engineering expense", The 27th Annual Conference of the IEEE Industrial Electronics Society

[9] Wu Sitao, Qian Qingquan, "Combing OPC woth Autonomous Decentralized Systems", 2000 IEEE

[10] Matthias Riedl, Mario Thron, Thomas Hadlich ," DriveServer-significantly reduce in engineering expens e" , IECON'01: The 27th Annual Conference of the IEEE industrial Electronics Society

[11] harald Kleines, "Access to Industrial Process Periphery Via java for Process Control(JFPC)" , IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL.49, NO.2, APRIL 2002

[12] IML, "Instrument Markup Language" , http://pioneer.gsfc.nasa.gov/public/iml/, 1999

[13] AIML, "Astronomical Instrument Markup Language" , http://pioneer.gsfc.nasa.gov/public/aiml/, 1999

본 연구는 한국과학재단 지정 울산대학교 네트워크기반 자동화연구센터 지원에 의해 이루어 졌습니다.