표준 XML 스키마 기반의 LD 그래픽 편집기 구현

권민혁°, 신승철
한국기술교육대학교 정보미디어공학과
{minhyuk, scshin}@kut.ac.kr

Graphic Editor for Ladder Diagrams based on PLCopen XML

Minhyuk Kwon°, Seungcheol Shin
Dept. of Information and Media Engineering. Korea University of Technology and Education

요 약

산업자동화, 임베디드시스템, 지능형 빌딩, 유비쿼터스 환경 구축 등의 다양한 분야에 활용되는 PLC나 모션제어기를 위한 표준안은 IEC 61131-3이다. 이 표준안은 제어기를 위한 프로그래밍 환경과 언어의 문법 구조를 정의하고 있으나, 도식적으로 표현되는 LD 프로그램의 저장 형식이 제시되지 않아서 관련 소프트웨어마다 서로 다른 저장 형식을 사용한다. PLCopen 그룹에서 배포한 표준 저장 형식 XML 스키마를 사용하면 데이터의 교환과 연동 언어들의 호환이 가능하다. 본 논문은 표준 XML 스키마를 기반하는 LD 그래픽편집기를 구현한다. 구현 형태는 Eclipse 플러그인으로서, Eclipse 도구인 EMF와 GEF를 이용하였다.

1. 서 론

PLC(Programmable Logic Controller)[1]는 기존의 물리적 회로에 의하여 작동하도록(Hardwired Logic) 만들어진 제어 장치를 메모리상의 프로그램(Softwired Logic)을 사용할 수 있도록 대체시킨 장치이다. PLC는 산업현장 전반에 걸쳐 사용하는데 특히, 산업자동화, 지능형 빌딩, 임베디드 시스템, 유비쿼터스 환경에 사용한다.

IEC 61131-3에는 PLC에서 사용할 수 있는 프로그래밍 언어 5가지를 정의한다[2]. 이 중 가장 많이 사용되는 프로그래밍 언어는 LD(Ladder Diagram)인데, 표준안에 정의된 LD 기호들을 이용하여 제어반내의 순차적인 제어를 도식적으로 표현하는 방식이다. 그러나 IEC 61131-3은 이런 도식적인 표현을 저장할 때 필요한 저장 형식을 제시하지 않고 있어서 개발환경마다 다양한 저장형식을 사용한다.

PLCopen그룹은 최근에 이런 점을 극복하기 위해 통합된 저장 방식인 XML 스키마를 발표하였는데, 이것은 LD 프로그램 뿐 아니라 FBD, IL, ST 등의 프로그램도 통합적으로 저장하고, 시스템 구성과 선언 사항들도 포함할 수 있게 설계되었다[4]. 이 XML 스키마를 저장 형식으로 사용하면 다른 개발환경과의 데이터 교환이 가능해 지고, IEC 61131-3에서 정의하는 언어들 사이의 변환성과 호환성이 향상된다.

본 논문에서는 PLCopen그룹의 TC6 XML 스키마를 기반으로 하는 LD 그래픽 편집기의 구현에 대하여 설명한다. LD 그래픽 편집기의 실제 구현은 Eclipse[8]의 EMF(Eclipse Modeling Framework)와 GEF(Graphical Editing Framework)[3]를 사용하여 Eclipse plug-in 형태로 이루어진다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2절은 기존의 개발 환경들과의 차이를 설명하고 3절은 구현된 LD 그래픽 편집기를 설명한다. 4절은 LD 그래픽 편집기의 형태와 출력 결과를 보여준다.

2. 관련 연구

LD 프로그래밍을 위한 개발 환경들은 많이 있지만 대부분은 IEC 61131-3의 요구만을 충족하고 PLCopen그룹의 TC6 XML 스키마와 같은 표준화된 저장 형식을 사용하지 않는다.

국내외에서 가장 많이 사용되는 제어 프로그래밍 개발환경으로는 ICS Triplex의 IsaGRAF[5]와 LS산전의 GMWIN[6]이 있는데, 이것들은 모두 고유의 저장 형식을 사용함으로써 다른 소프트웨어와 데이터 교환이 불가능하다.

PLCopen의 XML 스키마를 저장 형식으로 사용하는 개발환경으로는 KW-Software의 MULTIPROG[7]가 있는데, 이것은 독립형 개발도구이지만 본 논문의 그래픽 편집기는 Eclipse 플러그인으로 개발되었다.

본 논문에서 구현한 그래픽 편집기는 LD 프로그램을 PLCopen의 XML로 저장이 가능하고, 모든 LD 기호들은 자동으로 정확한 위치에 배치되며 연결선도 자동으로 그려진다. 또한, 본 논문의 그래픽 편집기는 Eclipse의 플러그인 형태로 개발이 되었기 때문에 확장이 수월하고, Eclipse에서 제공되는 많은 개발환경들을 이용하여 새로운 개발도구로의 변경이 가능하다. 또한 제공되는 다른 개발 환경의 요소와 쉽게 데이터 교환이 가능하다.

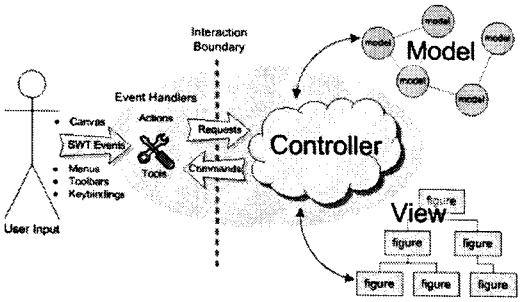
3. LD 그래픽 편집기 구현

여기서는 LD 그래픽 편집기를 구현하는 과정을 설명하는데, PLCOpen그룹에서 제공하는 TC6 XML 스키마를

저장 형식으로 사용하도록 하면서, Eclipse 기반의 플러그인 형태로 개발하는 과정을 나타낸다. 편집기 개발을 위해서 Eclipse의 모델링을 통합하기 위한 기술인 EMF와 MVC(Model-View-Controller)패턴 기반의 GEF 플러그인을 사용하였다. EMF를 이용하여 편집 대상인 LD 프로그램의 모델을 주어진 XML 스키마로부터 구성한다. 또한 GEF를 이용하여 그래픽 편집기를 설계한다.

3.1 LD 그래픽 편집기의 시스템 구성

LD 그래픽 편집기는 Eclipse기반으로 구현되었기 때문에 Eclipse에서 제공하는 도구들을 이용하여 개발을 할 수 있다. 편집기의 전체 구조는 GEF의 구조에 새로운 기능을 추가하여 구현되었다.



[그림 1. GEF 구조]

[그림 1]은 LD 그래픽 편집기 구현에 사용된 GEF의 구조이다. GEF는 MVC패턴으로 설계가 되어 있기 때문에 구현을 위해 TC6 XML 스키마를 EMF를 이용하여 모델을 운영하는 코드를 생성하였다. 화면에 보이는 부분을 담당하는 View는 GEF에서 제공하는 Draw2D 라이브러리를 이용하여 각 LD기호에 대응하는 그림을 그려주었다. LD에서는 접점과 출력, 전원선(PowerRail) 만을 다이어그램의 요소로 하고 있기 때문에, 간단하게 그릴 수 있다. 이렇게 제작된 모델과 뷰는 콘트롤러인 GEF의 EditPart 객체를 상속받은 클래스에게 둘 사이의 변화를 감지하여, 화면에 매순간 마다 표시 하여 사용자와 상호작용을 하게 된다.

LD 그래픽 편집기는 IEC 61131-3의 권고하는 방식으로 LD기호들을 배치한다. 때문에 이것을 위해 특별히 배치와 연결을 전담하는 클래스를 만들었고, 각 LD기호 객체를 식별하기 위해서 id를 중복되지 않게 할당하는 id 생성기를 만들었다. 또한 편집이 완료된 최종 모델을 XML 형태로 출력하기 위해서 EMF에서 제공하는 클래스를 이용하여 모델 전체를 저장하고 불러오는 기능을 하는 클래스를 만들었다. 이런 핵심 클래스에 대해서는 아래에서 설명한다.

3.2 모델의 생성과 수정

주어진 TC6 XML 스키마로 부터 모델을 생성할 때에는 스키마를 Eclipse에서 쓰는 메타-메타모델[1]로 변환한 후에 이것을 바탕으로 소스코드를 생성한다. 이 소스코드는 스키마에 맞도록 디자인된 자바 클래스형태의 모델인데, 이 클래스들을 적절히 조작함으로써 원하는 LD의 형태를 메모리상에 구축할 수 있다. LD 기호 객체를 생성할 때에는 직접 인스턴스를 만들 수도 있지만, 다른 클래스와 같이 생성된 Factory를 이용하여 생성하는 것이 일반적이다. 생성되는 모든 객체는 아무런 초기 값을 가지고 있지 않기 때문에, 스키마에 명시된 필수 요소들을 반드시 채운 후에 운영을 시작해야 한다. 그렇지 않으면 모델이 형태는 스키마에 부합하지만 모델의 검증과정에서 필수요소가 빠져 있는 것을 확인하고 다른 프로그램에서 잘못된 모델로 인식하게 된다.

```
public class ConstructTest {
  public static void main(String[] args) {
    Network network =
      ModelFactory.eINSTANCE.createNetwork();
    network.setNetworkName("PILab Network");
    System.out.println(network.getNetworkName());
  }
}
```

[그림 2. 모델의 생성 코드]

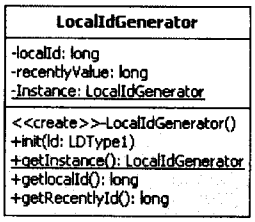
[그림 2]는 간단한 모델을 Factory로 부터 만들고 이것에 초기 값을 채워주는 프로그램이다. 모델운영 코드에 set으로 시작되는 메소드는 옵저버패턴[2]에 변경 사항을 통보하며, 이것은 EMF에서 제공하는 중요한 기능으로, 화면에 그려진 LD기호객체들이 사용자의 조작에 의해 위치 값이 변경 되었을 때 LD기호 역시 변경될 수 있도록 도와준다.

3.3 중요한 클래스들

GEF가 제공하는 기본 구조에 LD를 위한 편집기능을 추가하기 위해서는 가장 먼저 화면에 표시된 LD기호들의 배치를 결정하거나, 메모리에 존재하는 LD기호객체들의 id를 중복 없이 할당 하는 일은 매우 중요하다. id는 두 LD기호객체의 연결을 확인 할 수 있는 유일한 수단이기 때문이다. 이것을 위해서는 기본적으로 제공하는 도구 이외의 클래스를 디자인하고 구현할 필요가 있었는데, 여기에서 소개한다.

3.3.1 LocalIdGenerator LocalIdGenerator 클래스는 LD기호객체의 id 생성을 전담한다. [그림 3] 는 이 클래스의 다이어그램이다.
이 클래스는 싱글턴 패턴(Singleton Pattern)[3]을 기반으로

1) XML스키마를 모델을 표현하는 메타모델이라고 부르기 때문에 메타-메타모델로 부르는 것이 적당하다.
2) 옵저버(Observer)패턴은 주제 객체에 자식객체를 옵저버로 등록하고, 주제객체가 바뀔 때 자식객체에게 통보 되도록 디자인한 패턴이다.
3) 싱글턴 패턴(Singleton Pattern)은 public 생성자를 숨기고,



[그림 3. LocalIdGenerator의 클래스 다이어그램]

디자인 되어 있어서, id를 전담하는 객체가 프로그램에서 유일하게 하나만 생성됨을 보장한다.

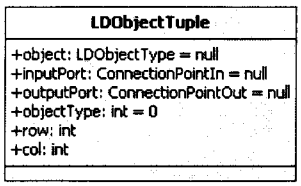
프로그램을 실행하면서 이 클래스의 인스턴스를 만드는 유일한 방법은, 최상위 LD기호객체를 이 클래스의 init 메소드의 인자로 넣고, getInstance() 메소드를 호출하는 방법이다. init 메소드가 실행 될 때, 입력으로 받은 최상위 LD기호객체의 자식 LD기호객체들을 모두 조사해서 가장 큰 id값 + 1을 생성하고, 다음 id 값을 만들 때마다 이 값은 1씩 증가된다. 이것으로 하나의 LD 모델에 유일한 id를 만들 수 있다. 이 클래스로 부터 새로운 id를 생성 받으려면, getLocalId() 메소드를 호출하여 long 타입의 정수 값을 얻을 수 있다. PLCOpen에서는 스키마에 id 값을 위해 unsigned long이라는 큰 자료형을 부여했지만, 사실 래더다이어그램에서 이렇게 많은 오브젝트가 나타나도록 디자인 하는 경우는 없다. 설사 이렇게 많은 LD기호 객체가 생성됨은 좋지 못한 프로그래밍을 했다는 의미이고, 이런 경우에는 예외를 발생하여 정상적으로 프로그래밍이 되지 않음을 알리도록 디자인 했다.

3.3.2 LadderLayoutManager 오브젝트 배치 문제를 해결하기 위해서 LD기호들은 항상 격자(Grid)형태로 배치된다는 점을 이용하여 사용자가 LD기호를 Viewer에 생성하고자 할 때 그 위치의 좌표 값을 격자의 눈금에 딱 맞는 값으로 임의로 조정하여 LD기호객체에 조정된 값을 넣었다. 이렇게 조정된 LD기호객체는 다른 LD기호객체들의 배치와 연결선 정보를 위해 배치형태와 연결선을 관리하는 LadderLayoutManager 클래스에게 작업이 위임된다.

이 클래스에서는 새로운 LD기호객체가 LD 최상위 객체에 추가될 때 이전의 상태를 2차원 배열에 기억하고 있다가 자식 LD기호객체를 추가할 때 유용하게 사용한다. LadderLayoutManager는 LD의 직선 연결과 OR 연결 그리고 분기된 연결에 대해서도 관리를 해주고 있으므로, 사용자의 편집형태에 따라 배열의 요소가 수시로 변화한다.

LadderLayoutManager 클래스도 위의 싱글턴 패턴으로 구현하려 했으나 GEF에서 제공하는 실행취소/다시하기 명령을 지원하기 위해 각 편집이 이루어지는 순간의 상태를 Command Stack에 기억할 필요가 있었다. 때문에 이 클래스는 싱글턴 패턴으로 구현하지 않고, 직접 인스턴스를 만들 수 있도록 디자인 하였다. 이 클래스에서

유일하게 하나의 객체의 생성만 보장하는 방법이다.



[그림 4. LDObjectTuple의 클래스 다이어그램]

지원하는 각 역할을 수행하기 위해서 2차원 배열에 저장되는 원소의 구조체를 정의할 필요가 있는데, 본 논문에서는 배열의 원소로 [그림 4]와 같은 구성을 선택하였다.

이 원소에는 현재 배치되는 오브젝트의 정보와 오브젝트의 연결을 위해 input/output port, 그리고 int 타입의 LD기호식별자를 두었고, 원소가 배열의 어느 위치에 저장되어 있는지를 판별하기 위해서 현재 원소가 위치하고 있는 배열의 행과 열의 정보도 같이 넣어 두었다. 이 정보를 기반으로 LadderLayoutManager클래스는 배치와 연결에 대한 모든 작업을 수행한다. 실제로 사용자가 LD기호를 Viewer에 생성하려고 할 때 이위치 값을 배열과 매핑하기 위해서 간단한 계산이 필요한데, 사용자가 LD기호중 접점기호를 (70,72) 위치에 놓는다면 이 정보는 격자에 맞게 다음과 같이 변환된 후에 새로운 LD기호객체에 저장된다.

$$x' = int(x / w) * w$$
$$y' = int(y / h) * h$$

이후, LD기호객체는 LadderLayoutManager에게 배치와 연결이 위임되는데, LD기호객체의 위치 값으로 2차원 배열에 넣어질 위치를 다음과 같이 알 수 있다.

$$col = x / w$$
$$row = y / h$$

위와 같이 LadderLayoutManager에서는 편집기의 가장 중요한 기능인 배치와, 다른 기호들과의 연결을 다루고 있긴 하지만 MVC패턴의 형태를 유지해야 한다. 따라서 LadderLayoutManager는 모델만을 수정하고, 이렇게 변경된 값을 콘트롤러에게 통보하여 뷰가 다시 그려질 수 있도록 유도한다.

```
public void save(IPath path) throws IOException {
  getResource(path);
  Map options = new HashMap();
  options.put(XMIResource.OPTION_DECLARE_XML, Boolean.TRUE);
  resource.save(options);
}
public void load(IPath path) throws IOException {
  getResource(path);
  Map options = new HashMap();
  options.put(XMIResource.OPTION_DECLARE_XML, Boolean.TRUE);
  resource.load(options);
}
```

[그림 5. 모델의 저장과 불러오기]

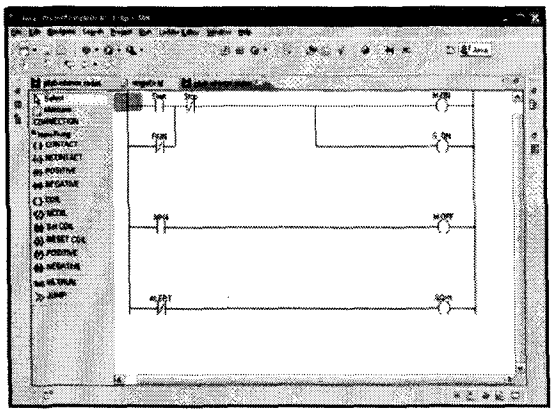
3.3.3 Tc6ModelManager Tc6ModelManager는 EMF에서 제공하는 클래스와 메소드를 모델을 편한게 다루도록 디

자인된 클래스이다. 이 클래스에서는 모델을 저장하거나 불러오는 것은 비교적 간단한다.

[그림 5]는 Tc6ModelManager의 저장과 불러오기를 구현한 완전한 메소드이다. 이 클래스는 실제 편집을 담당하는 SadariEditor 클래스에서 사용한다. SadariEditor에서는 모델을 전담하는 클래스를 별개로 두어 SadariEditor가 다른 형태로 변경될 때에도 재사용할 수 있다.

4. 구현 결과

본 논문에서 구현한 LD 그래픽 편집기는 Eclipse 기반의 플러그인들로 구현되었기 때문에, 편집기를 실행하기 위해서는 Eclipse 실행시간 프레임워크에 LD 그래픽 편집기를 설치해서 사용할 수 있다. [그림 6]는 구현된 편집기의 GUI의 모습과 여러 개의 네트워크를 갖는 예제 LD 프로그램이고 [그림 7]은 이것이 저장된 XML 파일의 일부만 첨부한 것이다.



[그림 6. 예제 LD 프로그램]

5. 결 론

본 논문은 IEC 61131-3의 LD 언어를 위한 그래픽 편집기 구현을 설명하였다. 구현된 LD 그래픽 편집기는 PLCopen XML 스키마를 저장 형식으로 채택하여 표준화된 데이터 교환이 가능하다. 또한 Eclipse 기반의 플러그인으로 구현함으로써 추후의 확장과 다른 개발 환경 요소들과의 연동이 수월하다.

현재 본 논문의 LD 그래픽 편집기에 LD-to-IL 컴파일러와 IL 인터프리터, IL 타입 시스템 등을 통합하는 작업을 진행 중이다.

참고 문헌

[1] R. R. Group and C, Engineering. Programmable logic controllers product research, 2006.

[2] IEC, International Standard IEC61131-3 Programmable controllers - Part 3: Programming languages 2nd Edition. International Electrotechincal Commission, 2003.

[3] William Moore et al, Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework, IBM Redbooks, 2004.


```
<?xml version="1.0" encoding="ASCII"?>
<tc6:LDType1 xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:tc6="http://www.plcopen.org/xml/tc6.xsd">
  <leftPowerRail height="352" localId="1" width="50">
    <position x="0" y="0"/>
    <connectionPointOut>
      <relPosition x="50" y="22"/>
    </connectionPointOut>
    ...생략...
  </leftPowerRail>
  <rightPowerRail height="352" localId="2" width="50">
    <position x="550" y="0"/>
    <connectionPointIn>
      <relPosition x="0" y="22"/>
      <connection refLocalId="7"/>
    </connectionPointIn>
    ...생략...
  </rightPowerRail>
  <contact variable="%IX1" height="32" localId="5" width="50">
    <position x="50" y="0"/>
    <connectionPointIn>
      <relPosition x="0" y="22"/>
      <connection refLocalId="1"/>
    </connectionPointIn>
    <connectionPointOut>
      <relPosition x="50" y="22"/>
    </connectionPointOut>
  </contact>
  <coil variable="%QX1" height="32" localId="8" width="50">
    <position x="500" y="320"/>
    <connectionPointIn>
      <relPosition x="0" y="22"/>
      <connection refLocalId="9"/>
    </connectionPointIn>
    <connectionPointOut>
      <relPosition x="50" y="22"/>
    </connectionPointOut>
  </coil>
  ... 생략 ...
</tc6:LDType1>
```

[그림 7. LD 프로그램의 XML 파일]

[4] PLCopen, Programmable Logic Controller open Group, http://www.plcopen.org.

[5] ICS triplex, ISaGRAF 5.0 제품 소개, http://www.isagraf.com/

[6] LS산전, GMWIN 제품 소개, http://www. lgis.lg.co.kr/product/product/introduce/detail_info.asp?p_id=P00201

[7] KW Software, MULTIPROG 제품 소개, http://www.kw-software.com/com/products/153.jsp

[8] Eclipse, Eclipse-an Open Development platform, http://www.eclipse.org