

사이버-물리 시스템의 ECML 을 위한 시물링크 XMI 변환기 개발

손현승*, 김우열**, 전인걸***, 전재호***, 김영철*

* 홍익대학교 컴퓨터정보통신공학과

세종특별자치시 조치원읍 세종로 2639

{son, bob}@selab.hongik.ac.kr

** 대구교육대학교 컴퓨터교육과

john@dnue.ac.kr

*** 한국전자통신연구원 CPS 연구팀

{igchun, jeonjaeho11}@etri.re.kr

요약: ECML 은 한국전자통신연구원에서 개발한 사이버-물리 시스템의 모델링 언어로 DEV&DESS 를 기반으로 이산과 연속 상태를 동시에 설계 가능한 하이브리드 모델링 언어이다. 그러나 ECML 은 현재 개발 중에 있기 때문에 아직 미완성으로 복잡한 수식 계산기능을 제공하지 못한다. 시물링크는 모델링, 시물레이션, 동적 시스템의 분석이 가능한 상용 도구로 ECML 과 같이 이산과 연속 상태의 모델링이 가능하고 다양한 도메인에서 사용 가능한 풍부한 라이브러리를 제공한다. 이러한 시물링크 모델을 사이버-물리 시스템의 모델링 언어인 ECML 에서 사용 가능하다면, ECML 의 단점을 보완 가능하다. 그래서 우리는 한국전자통신연구원과 함께 ECML 의 확장성을 높이기 위해 시물링크 모델을 연동하는 연구를 수행하고 있다. 그리고 시물링크 모델을 ECML 로 변환하기 위해서 메타모델을 기반한 모델변환 기법의 적용 방법을 제시하였다. 그러나 시물링크 모델 파일은 모델변환의 입력 파일인 XMI 가 아니기 때문에 바로 모델 변환에 적용하지 못한다. 본 논문에서는 시물링크 모델 파일을 모델변환에 사용하고자 XMI 변환기를 제안한다. 제안한 XMI 변환기는 C++로 작성되었고 3 가지 단계를 거쳐 시물링크 모델의 XML 파일을 XMI 파일로 자동 변환한다.

핵심어: 사이버-물리 시스템, 메타모델, 메타 모델 정의 언어(MOF), ETRI CPS Modeling Language (ECML), 시물링크

1. 서론

사이버-물리 시스템(Cyber-Physical Systems, CPS)은 대규모 센서/액추에이터를 가지는 물리적인 요소

와 이를 실시간으로 제어하는 컴퓨팅 요소가 결합된 복합시스템(System of Systems)이다[1]. 이외에 CPS는 다양한 정의들을 가지고 있는데, Edward Lee[2]는 “물리프로세스와 컴퓨팅의 결합체”, Wayne Wolf[3]는 “컴퓨터에 의해 제어되는 기계의 일종”, John Stankovic[4]는 “시스템의 동작이 모니터, 조율, 제어, 통합되는 물리적이고 엔지니어링된 시스템”이라고 언급하였다. 각 정의들이 조금씩 다르지만 컴퓨팅과 물리적인 시스템이 결합되어 있다는 공통점을 가지고 있다.

CPS는 새로 등장한 개념 같지만 기존의 임베디드 시스템으로부터 발전된 기술이다. 임베디드 시스템에 물리적 특성이 반영되고, 네트워크로 연결된 시스템 간의 제어의 중요성이 강조되는 시스템이다. 센서, 제어로직, 액추에이터를 가지는 대부분의 기존 제어 시스템들을 그대로 사용하지만 현재의 임베디드 시스템들은 예전에 비해서 훨씬 복잡해졌고 네트워크를 통해 상호 복합 연동되면서 시스템 설계시 인간의 논리력과 지능을 넘어서고 있다. 그러므로 CPS는 전통적인 임베디드 소프트웨어 개발 방법으로 다루기 어려운 이종 시스템들 간의 결합에 다른 복잡성, 불확실성, 불예측성 등을 추상화된 모델을 통해서 다루고 모델에 대한 검증은 포함하는 모델링&시물레이션이 확장된 개념이다.

CPS는 안전과 경쟁력을 유지시키는 주요 분야인 생물의학(Biomedical), 의료관리시스템(Healthcare System), 차세대 항공 교통 시스템(Next-Generation Air Transportation System), 스마트 그리드(Smart Grid), 재생 가능 에너지(Renewable Energy) 등의 다양한 분야에 적용가능하며 주요 기술로는 추상화(Abstract), 아키텍처(Architecture), 분산 컴퓨팅(Distributed Computations), 네트워크 제어(Networked Control), 검증 및 확인(Verification and

† 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업[10035708, 고신뢰 자율제어 SW를 위한 CPS(Cyber-Physical Systems) 핵심 기술 개발]과 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임.

Validation) 등이 있다[5].

ECML(ETRI CPS Modeling Language)은 한국전자통신연구원(ETRI)에서 개발중인 CPS 환경을 모델링하기 위한 언어이다[6]. ECML은 물리, 전기, 아날로그 전자 등과 같은 연속적인 요소와 디지털 전자, 소프트웨어 등과 같은 이산적인 요소를 모두 포함하는 하이브리드 시스템을 모델링할 수 있다[7]. 그러나 ECML은 현재 계속 개발 중에 있기 때문에 아직 미완성이고 복잡한 수식 계산기능을 제공하지 못한다.

시물링크는 모델링, 시물레이션, 동적 시스템의 분석이 가능한 상용 도구이다. 다양한 도메인에서 사용 가능한 풍부한 라이브러리(이벤트 기반 모델링, 물리적 모델링, 제어 시스템 설계 및 분석, 신호 처리 및 통신, 코드 생성, 검증 및 테스트)를 제공하기 때문에 많은 분야(항공 우주 및 국방, 자동차, 생명 과학 및 제약, 통신, 전자 및 반도체, 에너지, 금융 서비스, 산업 자동화 및 기계)의 공학자들에게 사용되고 있다. 그리고 대화식 그래픽 환경 및 사용자 정의 가능한 블록 라이브러리 세트를 제공하고 특화된 애플리케이션으로의 확장이 가능하다. 또한 가상 시제품을 빠르게 제작, 테스트함으로써 최소의 노력으로 어떤 항목의 단계에서라도 디자인 컨셉을 확정할 수 있도록 지원하여, 많은 엔지니어들이 시제품 제작 전 시물링크를 사용하여 반복 동작과 설계를 수행한다[8].

이러한 시물링크 모델을 CPS의 모델링 언어인 ECML에서 사용 가능하다면, ECML의 단점을 보완 가능하다. 그래서 우리는 한국전자통신연구원과 함께 ECML의 확장성을 높이기 위해 시물링크 모델과 연동에 관한 연구를 수행하고 있다[9]. 그리고 시물링크 모델을 ECML로 변환하기 위해서 메타모델을 기반한 모델변환 기법의 적용 방법을 제시하였다[10]. 모델 변환은 이종의 서로 다른 모델을 메타모델을 기반으로 모델에서 모델로 변환시켜줄 수 있는 기법이다[11]. 그러나 시물링크 모델 파일은 모델변환의 입력 파일인 XMI(XML-based Metadata Interchange)[12]가 아니기 때문에 바로 모델 변환에 적용하지 못하는 문제점을 가지고 있다.

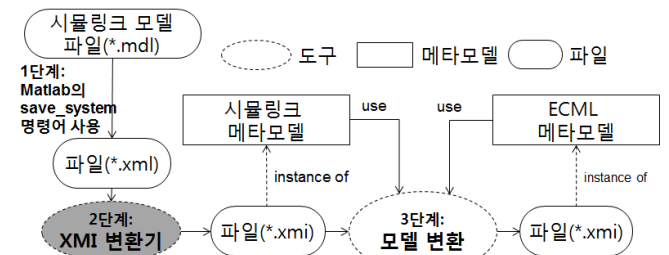
본 논문에서는 시물링크 모델 파일을 모델변환에 사용할 수 있도록 시물링크 모델의 XML 파일을 XMI로 변환할 수 있는 XMI 변환기를 제안한다. 제안한 XMI 변환기는 C++로 작성되었고 3가지 단계를 거쳐 시물링크 모델의 XML 파일을 XMI 파일로 자동 변환한다. 첫 번째 단계는 불필요한 시물링크 데이터를 제거한다. 두 번째 단계는 시물링크의 XML 파일의 값을 모두 속성으로 변환한다. 세 번째는 메타모델의 네임스페이스를 입력한다. 적용사례를 통해서 시물링크 모델을 생성하고 XMI 변환기로 변환한다. 이러한 단계를 거쳐 만들어진 XMI가 모델 변환에서 사용가능한지 EMF(Eclipse Modeling Framework)[13]를 통해서 파일을 읽어 검증해본다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 모델변환을 이용한 시물링크에서 ECML 모델

변환방법에 대해서 설명한다. 3장에서는 제안한 XMI 변환기에 대해서 언급한다. 4장에서는 적용사례로 시물링크 모델로부터 XMI를 생성해보고 이를 EMF로 검증한다. 마지막으로 5장에서는 결론과 향후 연구에 대해서 기술한다.

2. 관련연구

시물링크 모델로부터 ECML로의 모델 변환하는 방법은 [그림 1]과 같이 3단계의 절차가 필요하다. 먼저 첫 번째 단계에서는 Matlab의 save_system 명령어를 사용하여 시물링크 MDL 파일을 XML로 변환한다. 두 번째 단계에서는 XMI 변환기를 통해서 시물링크 XML 파일을 XMI로 변환한다.



[그림 1] 시물링크 모델로부터 ECML로의 변환 방법[10]

세 번째 단계에서는 XMI 파일로 변경된 시물링크 모델 정보를 ECML 구조로 모델 변환 기법을 사용하여 변환한다. 그러나 모델 변환 기법을 사용하기 위해서는 메타모델이 설계 되어 있어야 하는데, 시물링크 메타모델[10]과 ECML 메타모델[6]은 기존 연구에서 설계하였다. 그리고 모델변환 단계에서는 시물링크와 ECML 모델과의 공통점과 차이점을 분석하여 변환 규칙을 만든다. 이 부분은 향후 연구과제이다. 본 논문의 XMI 변환기는 모델 변환을 수행하는 전체과정 중에서 우선적으로 수행되어 할 부분이며 그 위치는 [그림 1]에 음영으로 표현하였다.

3. XMI 변환기 개발

XMI 변환기의 목적은 기존의 시물링크의 XML 파일에서부터 필요한 정보만 추출하여 모델 변환의 입력파일을 만드는 것이다. 그전에 MDL 파일로부터 XML 파일로 변환해야 한다. 이 변환 방법은 Matlab에서 제공하기 때문에 이를 그대로 사용한다. MDL 파일을 XML로 만들기 위해서는 “save_system” 명령을 수행해야 한다. 명령어의 형식은 아래와 같다.

```

save_system('MDL 파일 이름',
'출력할 XML 파일 이름', 'ExportToXML', true)
    
```

본 장에서는 시물링크 XML 파일 구조를 분석하고

XMI 변환기의 변환방법에 대해서 설명한다.

3.1 시뮬링크 XML 파일 구조

시뮬링크의 XML 파일의 전체적인 구조는 [그림 2]와 같다. 루트는 ModelInformation 으로 Version 정보를 포함하고 있다. 그리고 Model 과 Stateflow 로 구성되는 Model 은 시뮬링크 모델과 관련된 정보를 다루고 있고 Stateflow 는 Statechart 와 관련되어 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
- <ModelInformation Version="0.9">
- <Model Name="my_system">
  <P Name="Version">7.4</P>
  <P Name="MdlSubVersion">0</P>
  + <GraphicalInterface>
    <P Name="SavedCharacterEncoding">windows-949</P>
    <P Name="SaveDefaultBlockParams">on</P>
    <P Name="ScopeRefreshTime">0.035000</P>
    <P Name="OverrideScopeRefreshTime">on</P>
    <P Name="DisableAllScopes">off</P>
    <P Name="DataTypeOverride">UseLocalSettings</P>
    <P Name="MinMaxOverflowLogging">UseLocalSettings</P>
    <P Name="MinMaxOverflowArchiveMode">Overwrite</P>
    <P Name="MaxMDLFileLength">120</P>
  + <ConfigManagerSettings>
  + <EditorSettings>
  + <SimulationSettings>
  + <Verification>
  + <ExternalMode>
  + <EngineSettings>
  + <ModelReferenceSettings>
  + <ConfigurationSet>
  + <SystemDefaults>
  + <BlockDefaults>
  + <AnnotationDefaults>
  + <LineDefaults>
  + <BlockParameterDefaults>
  + <System Name="my_system">
  + <Model>
  + <Stateflow>
</ModelInformation>
```

[그림 2] 시뮬링크 XML 파일의 내용

Model 의 내용을 살펴보면 순서대로 GraphicalInterface, ConfigManagerSettings, EditorSetting, SimulationSetting, Verification, ExternalMode, EnginSetting, ModelReferenceSettings, Configuration-Set, SystemDefaults, BlockDefaults, Annotation-Defaults, LineDefaults, BlockParameterDefaults, System 으로 구성되어 있다. 위에 보는 것과 같이 많은 정보를 포함하고 있다.

GraphicalInterface 는 그래픽 인터페이스 정보를 저장하는 항목으로 내용은 아래와 같다. 시뮬링크 내부에서 사용하는 정보이기 때문에 무엇을 뜻하는지 알 수 없다. ConfigManagerSettings 은 파일을 만든 사용자 정보가 저장된다. EditorSetting 은 시뮬링크 모델을 그리는 에디터에 대한 설정 값을 포함한다. SimulationSetting 은 시뮬레이션을 수행하기 위한 설정 값을 저장한다. Verification, ExternalMode, EnginSetting, ModelReferenceSettings, Configuration-Set 들은 시뮬링크 내부적으로 사용되는 값이기 때문에 어떤 정보인지 알 수 없다. SystemDefaults 는 시뮬링크 모델링하는 작업창의 크기 및 그래픽 정보를 저장한다. BlockDefaults 는 시뮬링크에서 사용되는 각 블록들에 대한 기본적인 그래픽 정보(폰트, 색깔, 크기 등)을 저장한다. Annotation- Defaults 는 설명 글을 사용할 때 사용하는 기본적인 그래픽 정보

를 저장한다. LineDefaults 는 선에 대한 그래픽 정보를 저장한다. BlockParameterDefaults 는 블록에서 사용하는 기본적인 파라미터정보를 저장한다. 이것은 블록에서 필요한 파라미터를 사용자가 수작업으로 모두 입력하지 않아도 되므로 편하게 모델링이 가능하다. [그림 3]과 같이 만들어진 블록에 대한 파라미터 정보를 저장한다.

```
- <BlockParameterDefaults>
  + <Block BlockType="Gain">
  + <Block BlockType="Inport">
  + <Block BlockType="Integrator">
  + <Block BlockType="Output">
  + <Block BlockType="Saturate">
  + <Block BlockType="Scope">
  + <Block BlockType="Sin">
  + <Block BlockType="Subsystem">
  + <Block BlockType="ZeroOrderHold">
</BlockParameterDefaults>
```

[그림 3] BlockParameterDefaults 파일의 내용

System 은 시뮬링크의 블록에 대한 정보가 저장되는 영역이다. 각 블록정보와 라인 정보들을 [그림 4]와 같이 확인할 수 있다.

```
- <System Name="my_system">
  <P Name="Location">[279, 261, 942, 766]</P>
  <P Name="Open">on</P>
  <P Name="ModelBrowserVisibility">off</P>
  <P Name="ModelBrowserWidth">200</P>
  <P Name="ScreenColor">white</P>
  <P Name="PaperOrientation">landscape</P>
  <P Name="PaperPositionMode">auto</P>
  <P Name="PaperType">A4</P>
  <P Name="PaperUnits">centimeters</P>
  <P Name="TiledPaperMargins">[1.270000, 1.270000, 1.270000, 1.270000]</P>
  <P Name="TiledPageScale">1</P>
  <P Name="ShowPageBoundaries">off</P>
  <P Name="ZoomFactor">100</P>
  <P Name="ReportName">simulink-default.rpt</P>
  <P Name="SIDHighWatermark">18</P>
  + <Block Name="Gain" BlockType="Gain">
  + <Block Name="Integrator" BlockType="Integrator">
  + <Block Name="Saturation" BlockType="Saturate">
  + <Block Name="Scope" BlockType="Scope">
  + <Block Name="Sine Wave" BlockType="Sin">
  + <Block Name="Subsystem" BlockType="Subsystem">
  + <Line>
  + <Line>
  + <Line>
  + <Line>
  + <Line>
  + <Line>
  + <Annotation>
  + <Annotation>
</System>
```

[그림 4] System 파일의 내용

우리는 시뮬링크 XML 파일 구조 분석을 통해서 필요한 정보와 그렇지 않은 정보들을 구분하였다. 필요한 정보는 시뮬링크의 블록을 저장하고 있는 System 항목과 파라미터의 기본 정보를 포함하는 BlockParameterDefaults 항목이다. 그러므로 이를 선 처리해주는 프로그램이 필요하다. 그리고 모든 영역에 P 엘리먼트가 존재하는데 P 엘리먼트의 속성은 Name 과 Value 두 가지 형태로 구성되어 있다. 시뮬링크 XML 은 모델의 정보를 만들 때 각 항목의 속성 값으로 데이터를 저장하지 않고 여러 개의 P 엘리먼트를 이용하여 저장하는 특징을 가지고 있다. 그 외의 나머지는 항목들은 모두 시뮬링크가 시뮬레이션을 수행하기 위한 정보들로 제 3자가 이해하기 어려운 정보들로 구성되어 있고 모델 변환 시에도 필요가 없다. 그러므로 제거 되어야 할 항목들 이다.

3.2 시물링크 XML 에서 XMI 로의 변환 방법

시물링크 XML 파일 구조 분석결과 System 과 BlockParameterDefaults 항목만 필요하고 나머지 정보는 필요 없다는 것을 알 수 있었다. 수작업으로 이를 처리할 수도 있지만 비생산적인 일이기 때문에 XMI 변환기를 개발하였다. XMI 변환기는 두 가지 목적으로 만들어 졌다. 첫 번째는 위에서 언급하였듯이 시물링크 XML 파일에서의 필요한 정보의 추출이다. 두 번째는 XML 과 XMI 의 차이점 때문이다. 모델 변환을 수행하기 위해서는 모델변환 도구가 인식 가능한 XMI 형태로 변환되어야 하기 때문이다.

3.2.1 시물링크 XML 에서 필요한 정보 추출 방법

시물링크 XML 에서부터 필요한 정보 추출 방법은 알고리즘은 두 단계로 이루어진다. 첫 번째로는 한 정보를 추출하기 위해서 System 과 BlockParameterDefaults 이외의 필요 없는 항목 정보를 제거 한다. 두 번째는 필요 없는 P 항목을 제거이다.

```
RemoveNode("//Model/GraphicalInterface");
RemoveNode("//Model/ConfigManagerSettings");
RemoveNode("//Model/EditorSettings");
RemoveNode("//Model/SimulationSettings");
RemoveNode("//Model/Verification");
RemoveNode("//Model/ExternalMode");
RemoveNode("//Model/EngineSettings");
RemoveNode("//Model/ModelReferenceSettings");
RemoveNode("//Model/ConfigurationSet");
RemoveNode("//Model/SystemDefaults");
RemoveNode("//Model/BlockDefaults");
RemoveNode("//Model/AnnotationDefaults");
RemoveNode("//Model/LineDefaults");
RemoveNode("//ModelInformation/Stateflow");
```

[그림 5] 첫 번째 단계의 불필요한 항목 제거

첫 번째 단계의 항목 정보 제거는 [그림 5]와 같이 C++언어로 작성되었고 XML 의 XPATH 를 사용하여 항목을 찾고 이를 제거한다. 두 번째 단계의 P 항목을 제거는 [그림 6]과 같이 C++언어로 작성되었다. P 항목은 모든 영역에 분포되어 있으므로, 첫 번째 단계의 항목제거와 같이 하위항목을 전부 지우면 안 된다. 또한 Model 이나 System 항목은 제거하면 안 되기 때문에 두 번째 인자를 활용해서 항목의 이름이 같지 않을 때만 제거하는 함수를 새로 만들었다. 이 RemoveAllNotMatchNodesName 함수는 두 번째 인자의 이름과 매치되지 않는 P 항목을 모두 제거한다. 그리고 여러 개를 동시에 선택할 수 있도록 “ ; ” 구분자를 사용하도록 하였다. 예를 들어 “Version;MdlSubVersion”는 P 항목의 이름이 “Version” 과 “MdlSubVersion” 만 제외하고 모두

제거한다.

```
RemoveAllNotMatchNodesName("//Model/P",
"Version;MdlSubVersion");
RemoveNode( "//Model/System/P");
RemoveAllNotMatchNodesName(
"//System/Annotation/P", "Name");
```

[그림 6] 두 번째 단계의 불필요한 항목 제거

3.2.2 XML 에서 XMI 형태로 변환 방법

XML 파일을 XMI 형태로 변환하기 위해서는 두 단계가 필요하다. 첫 번째는 XML 의 Value 를 속성 값으로 변경하는 작업이다. XMI 는 Value 를 사용하지 않고 속성만 사용하기 때문이다. 두 번째는 메타 모델과의 연동을 위한 루트 노드에 네임스페이스를 삽입해야 한다.

```
MakeValue("//Model/P");
MakeValue("//Block/P");
MakeValue("//Port/P");
MakeValue("//System/P");
MakeValue("//List/P");
MakeValue("//Line/P");
MakeValue("//Branch/P");
MakeValue("//Annotation/P");
```

[그림 7] Value 를 속성 값으로 변경

첫 번째 단계의 XML 의 Value 를 속성 값으로 변경하는 방법은 [그림 7]과 같다. MakeValue 함수는 해당되는 항목을 찾아서 Value 를 속성으로 변경한다. 이 코드를 수행하면 [그림 8]과 같이 Value 가 속성값으로 변경된 결과를 확인할 수 있다.

변환 전
<pre><Block BlockType="Gain"> <P Name="Gain">1</P> <P Name="Multiplication">E-w(K.*u)</P> <P Name="ParamMin">[]</P> <P Name="ParamMax">[]</P> ...(<생략>) </Block></pre>
변환 후
<pre><Block BlockType="Gain"> <P Name="Gain" Value="1"/> <P Name="Multiplication" Value="E-w(K.*u)"/> <P Name="ParamMin" Value="[]"/> <P Name="ParamMax" Value="[]"/> ...(<생략>) <Block></pre>

[그림 8] Value 를 속성 값으로 변경 실행 결과

두 번째 단계의 메타모델과의 연동을 위한 루트 노드에 네임스페이스를 삽입 방법은 [그림 9]와 같다.

```
MakeNamespace("simulink",
"http://simulink/1.0");
```

[그림 9] 네임스페이스를 삽입 방법

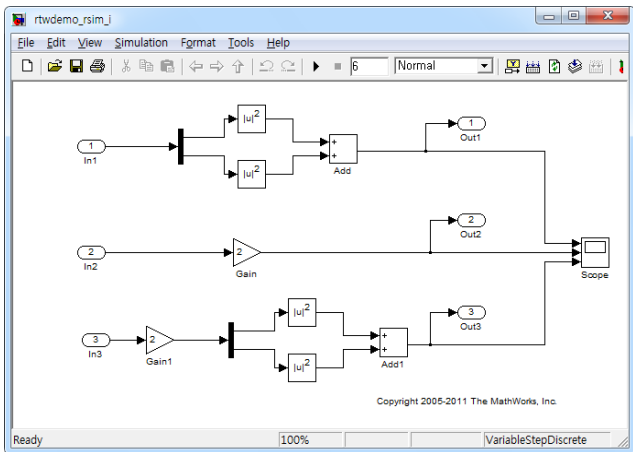
입력되는 네임스페이스는 메타모델의 네임스페이스와 같은 값으로 설정해야 된다. 실행 결과는 [그림 10]과 같다.

변환 전
<ModelInformation Version="0.9"> ...(생략) </ModelInformation>
변환 후
<simulink:ModelInformation xmlns:simulink="http://simulink/1.0" Version="0.9"> ...(생략) </simulink:ModelInformation>

[그림 10] 네임스페이스를 삽입 방법의 실행 결과

4. 적용사례

시물링크 모델의 XMI 생성을 위해서 [그림 11]과 같이 시물링크에 도움말에서 기본으로 제공되는 Rapid Simulation 예제를 사용하였다.



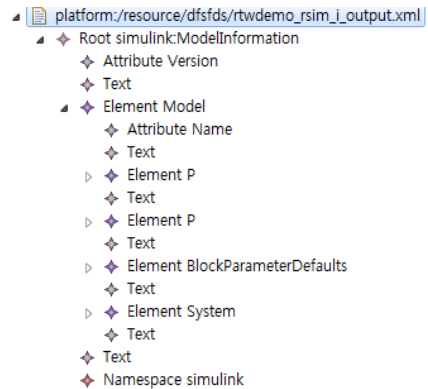
[그림 11] 시물링크 Rapid Simulation 모델링 예제

이 예제를 XMI 변환기를 실행하면 [그림 12]와 같이 XMI 파일이 생성된다.

```
<?xml version="1.0" encoding="UTF-8"?>
- simulink:ModelInformation Version="0.9" xmlns:simulink="http://simulink/1.0">
- <Model Name="rtwdemo_rsim_i">
  <P Name="Version" Value="7.8"/>
  <P Name="MdlSubVersion" Value="0"/>
  + <BlockParameterDefaults>
  - <System>
    + <Block Name="In1" BlockType="Inport" SID="1">
    + <Block Name="In2" BlockType="Inport" SID="2">
    + <Block Name="In3" BlockType="Inport" SID="3">
    + <Block Name="Add" BlockType="Sum" SID="4">
    + <Block Name="Add1" BlockType="Sum" SID="5">
    + <Block Name="Demux" BlockType="Demux" SID="6">
    + <Block Name="Demux1" BlockType="Demux" SID="7">
    + <Block Name="Gain" BlockType="Gain" SID="8">
    + <Block Name="Gain1" BlockType="Gain" SID="9">
    + <Block Name="Math Function" BlockType="Math" SID="10">
    + <Block Name="Math Function1" BlockType="Math" SID="11">
    + <Block Name="Math Function2" BlockType="Math" SID="12">
    + <Block Name="Math Function3" BlockType="Math" SID="13">
    + <Block Name="Scope" BlockType="Scope" SID="14">
    + <Block Name="Out1" BlockType="Outport" SID="15">
    + <Block Name="Out2" BlockType="Outport" SID="16">
    + <Block Name="Out3" BlockType="Outport" SID="17">
  + <Line>
  ... (생략)
  + <Annotation>
  </System>
</Model>
</simulink:ModelInformation>
```

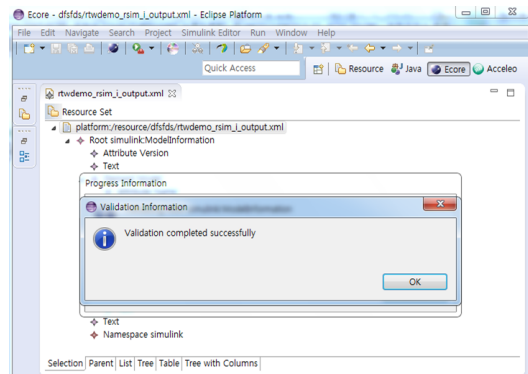
[그림 12] XMI 변환기로 생성한 XMI 파일

생성된 XMI의 검증을 위해서 [그림 13]과 같이 EMF를 이용하여 XMI 파일을 읽었다. 그 결과 에러 없이 XMI 파일을 읽을 수 있었다.



[그림 13] EMF를 이용하여 생성된 XMI 파일 로드

[그림 14]는 EMF가 제공하는 검증기능을 수행하였을 때 실행된 결과이다. EMF의 검증기능은 메타모델을 기반으로 잘못된 데이터가 없는지 체크한다. 검증 수행결과 성공 메시지를 확인할 수 있었다.



[그림 14] EMF를 이용하여 생성된 XMI 검증결과

5. 결론

ECML 은 CPS 환경을 모델링하기 위한 언어로 연속적인 요소와 이산적인 요소를 모두 포함하는 하이브리드 시스템을 설계할 수 있다. 그러나 ECML 은 현재 계속 개발 중에 있기 때문에 아직 미완성이고 복잡한 수식계산 기능을 제공하지 못한다. 이에 비하여 시물링크는 모델링, 시물레이션, 동적시스템의 분석이 가능한 상용 도구로 다양한 도메인에서 사용가능한 풍부한 라이브러리를 제공하며 1984 년부터 현재 동안 오랜 기간 동안 개발되어진 매우 안정된 제품이다.

이러한 시물링크 모델을 ECML 에 적용 가능하다면 ECML 의 단점을 보완가능하고 확장성도 제공할 수 있는 장점이 있다. 이러한 이유로 시물링크 모델을 ECML 에 연동하기 위한 방법을 연구하고 있다. 본 논문에서는 시물링크 모델 파일이 모델변환의 입력 파일인 XMI 에 적용하지 못하는 문제를 해결하고자 XMI 변환기를 제안하였다.

제안한 XMI 변환기는 C++로 작성되었고 3 가지 단계를 거쳐 시물링크 모델의 XML 파일을 XMI 파일로 자동 변환이 가능하였다. 그리고 적용사례를 통해서 시물링크 모델을 XMI 변환기로 변환하고 EMF 를 통해서 검증하였다. 검증 결과 XMI 변환기를 사용하면 시물링크 모델을 모델 변환에 사용할 수 있음을 확인하였다.

향후 연구로는 최종연구의 나머지 부분인 시물링크 모델을 ECML 로 변환하기 위한 규칙들을 만들고 EMF 를 이용하여 모델변환 도구를 개발하는 것이다. 현재 XMI 변환기를 통해 생성된 XMI 파일을 이용하여 ECML 로 변환하기 위한 방법을 연구 중이다.

참고문헌

- [1] 김원태, 전인걸, 이수형, 박승민, "CPS 기술 동향," 정보통신산업진흥원 주간기술동향, 통권 1455 호, pp. 1 - 10, 2010.07.21.
- [2] Edward A. Lee, "Cyber Physical Systems: Design Challenges," Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on, pp. 363-369, 2008. 05.
- [3] Wayne Wolf, "Cyber-physical Systems," Embedded Computing, pp. 88-89, 2009. 03.
- [4] R. Rajkumar, I. Lee, L. Sha, J. Stankovic, "Cyber-physical systems: the next computing revolution," Proceedings of the 47th Annual Design Automation Conference (DAC '10), pp. 731-736, 2010. 06.
- [5] K. Baheti and H. Gill, "Cyber-physical systems," The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), IEEE Control

Systems Society, 2011.

- [6] Jaeho Jeon, InGeol Chun, WonTae Kim : Metamodel-Based CPS Modeling Tool. Embedded and Multimedia Computing Technology and Service, Lecture Notes in Electrical Engineering Vol. 181, 285--291 (2012)
- [7] Hae Young Lee, Ingeol Chun, Won-Tae Kim : DVML: DEVS-Based Visual Modeling Language for Hybrid Systems. Control and Automation, and Energy System Engineering, Communications in Computer and Information Science Vol. 256, 122--127(2011)
- [8] MATLAB/Simulink, <http://www.mathworks.com>
- [9] ETRI, Cyber-Physical Systems research, http://www.etri.re.kr/etri/res/res_05020102.etri
- [10] Hyun Seung Son, Woo Yeol Kim, Robert Young Chul Kim, Hang-Gi Min, "Metamodel Design for Model Transformation from Simulink to ECML in Cyber Physical Systems," Computer Applications for Graphics, Grid Computing, and Industrial Environment, CCIS 351, pp. 56-60, 2012.12.17.
- [11] K. Czarnecki, S. Helsen, "Feature-based survey of model transformation approaches," IBM Systems Journal, Vol. 45, Issue 3, pp. 621-645, 2006.
- [12] OMG, MOF 2.0/XMI Mapping, version 2.1.1, formal/2007-12-01
- [13] D. Steinberg, F. Budinsky, E. Merks, M. Paternostro, "EMF: eclipse modeling framework," Addison-Wesley Professional, 2008.