데이터 흐름 언어로 구현되는 시스템 모델링을 위한 UML 확장

심재근°, 이종원, 이병정*, 우치수
서울대학교 컴퓨터공학부, * 서울시립대학교 컴퓨터과학부
{jkshim°, ljw, wuchisu}@selab.snu.ac.kr, bjlee@venus.uos.ac.kr*

Extending UML for Modeling Systems in Data Flow Languages

Jaekeun Shim°, Chongwon Lee, Byungjeong Lee*, Chisu Wu
School of Computer Science and Engineering, Seoul National University
* School of Computer Science, University of Seoul

요 약

데이터의 흐름과 그 데이터의 처리가 중요한 시스템이 있다. 이러한 시스템을 구현하는 데이터 흐름 언어들은 데이터 흐름도로 모델링하는 것이 가장 이해하기 쉽고 정보의 손실이 적은 방법이다. 이러한 시스템은 분산 환경의 중요성이 높아짐에 따라 다른 이종적인 시스템과 같이 개발 될 가능성이 높아지고 있다. 따라서 이종적인 시스템에 대한 일관성 있고 통합적인 설계와 문서화가 필요하게 되었다. UML은 객체 지향 언어로 구현되는 시스템의 모델링과 문서화에 있어서는 사실상의 표준이지만 데이터 흐름의 구성 요소들이 객체 지향적이지 않기 때문에 UML로 데이터 흐름을 나타내는 것은 어렵다. 따라서 우리는 이러한 문제를 해결하기 위한 UML 확장 메커니즘을 제안한다. 데이터 흐름 언어가 객체와 클래스로 사상될 수 있는 특성들을 가지고 있다면 그 특성들을 UML 다이어그램으로 확장할 수 있다. 그러기위해 새로운 스테레오타입들을 정의하여 기존의 UML 다이어그램 구성물과의 차이를 둔다. 이러한 과정을 통해 UML 사용자들은 데이터 흐름 언어의 구성 요소들을 캡슐화 할 수 있다. 데이터 흐름 언어로 구현되는 일반적인 시스템에 적용하기 위해 우리는 UML 메타모델 개요를 제안하는 것으로 스테레오타입들을 정의한다. 그리고 이렇게 확장된 UML 메커니즘을 가지고 데이터 흐름 언어로 구현된 두 개의 시스템을 설계하고 설계에 따라 구현 하였다.

1. 서 론

데이터 흐름 언어(data flow language)는 시각적 프로그래밍 언어(visual programing language)로서 데이터 흐름 구조나 아키텍쳐 또는 프로그램의 모델을 데이터의 방향성 있는 그래프로 구현한다. 데이터 흐름을 나타내는 설계 방식은 기계 회로 등의 하드웨어의 설계에 많이 사용되었고, 하드웨어 제어 시스템은 이러한 데이터 흐름 언어로 구현되는 경우가 많다. 이전부터 컴퓨터 비전(computer vision) 등 데이터와 그 처리 흐름이 중요한 프로그램 개발에 있어서는 데이터 흐름 언어가 주로 사용되어 왔다[1]. 이런 언어들은 단순한 구조를 가지고 있으나 개발 시스템의 목적에 맞게 오랜 기간 발전해왔고, 데이터와 연산을 구현하는데 효율적이고 직관적으로 구성되어있다.

데이터 흐름 언어의 데이터와 데이터의 흐름을 모델링 하는데 있어서 데이터 흐름도(data flow diagram)는 가장 직관적이고 쉬운 기술 방법 중 하나이다. 또한 시스템의 데이터의 흐름에 대한 모델링에는 흐름 도표(flow chart)도 많이 사용되었다[2].

한편, 시스템이 점점 커지고 복잡해짐에 따라 분산 환경과 분산 처리의 중요성이 점점 커지고 있다. 과거에는 하드웨어의 제어 프로그램이 개별적으로 기능했지만 이제는 다른 응용 프로그램과 같이 개발되는 경우가 점점 늘어나고 있다. 이와 함께 시스템을 구현하는 언어 역시 다양하고 이종적인 언어로 구성될 가능성이 커진다. 따라서 이종적인 언어들로 구현되는 각각의 하위 시스템을 아우르는 일관적이고 통합적인 설계는 중요하다[3].

UML[4]은 객체 지향 언어의 설계에 최적화되어있는 모델링 언어로 객체 지향 언어로 구현되는 시스템 설계와 문서화와 있어서는 가장 널리 사용되는 표준이다. 이런 점에서 이종적인 시스템들의 총체적이고 일관된 설계와 문서 관리를 위해서 UML을 이용하는 것은 효율적이다. 하지만 UML은 데이터 흐름도나 데이터 흐름 언어를 효과적으로 모델링 하는 다른 표기법을 가지고 있지 않다.

이 문제를 해결하기 위해 내장형(embeded) 시스템의 모델링에 있어서 교류도(collaboration diagram)를 사용하여 데이터 흐름을 UML로 표현한 연구가 있다[5]. 그렇지만 그 외에 데이터 흐름을 UML로 표현하는 연구는 적었다. 이는 데이터 흐름을 UML로 표시하기에는 데이터 흐름을 나타내는 구성 요소가 비 객체 지향적이기 때문이다. 본 연구에서는 더욱 효율적으로 데이터 흐름도

를 UML로 표현하기 위한 UML 확장 메커니즘을 제시한다. 그리고 이를 적용하여 실제 시스템을 설계하여 성공적으로 개발한 사례를 소개한다.

이 논문은 다음과 같이 구성되어 있다. 2장에서는 관련된 다른 선행 연구들을 소개하고 3장에서는 UML 확장의 기반이 되는 확장된 메타모델과 확장 원칙들을 제시한다. 4장에서는 3장에서 제시된 내용을 적용하여 설계하고 구현한 사례 연구를 소개한다. 그리고 결론과 향후 연구에 대한 언급은 5장에서 한다.

2. 관련 연구

UML은 객체 지향 언어의 모델링에 있어서는 사실상의 표준이다. 하지만 데이터 흐름을 나타내는데 중요한 요소인 데이터나 데이터의 흐름, 데이터의 연산 등을 잘 나타낼만한 표현 방법은 마땅히 없다. 그런 이유로 기존의 UML을 통한 데이터 흐름 언어의 모델링은 그다지 고려되지 않았다. 그렇지만 데이터 흐름을 객체 지향적으로 사상하여 표현하고자 하는 선행된 연구는 많지 않았다.

UML로 거대하고 복잡한 시스템을 객체 지향적인 방법보다 효과적으로 설계하기 위해 기능 클래스 분해(Function-Class Decomposition)가 제안 되었다[6]. 기능 클래스 분해 방법을 통해 시스템을 모델링하는 과정은 다음과 같다. 먼저 크고 복잡한 시스템 기능을 위주로 최소의 기능적 기본 단위(functional module)를 얻을 때까지 쪼갠다. 이러한 분해 과정을 통해 얻은 기능적 기본 단위를 UML의 클래스로 사상한다. 그 다음 사상된 클래스간의 상호작용을 나타내는 클래스를 구현하여 각 기능적 기본 단위 간의 관계를 설계한다. 기능 클래스 분해 방법을 통해 이종적인 언어로 구현되는 시스템을 설계하는 것이 가능하다. 그러나 하드웨어 제어 프로그램 등의 데이터 흐름과 처리가 중요한 하위 시스템에서 기능적 단위로 쪼갠다고 해서 이러한 기능적 단위가 클래스로 바로 사상될 수는 없다. 더욱이 클래스간의 상호작용을 나타내는 클래스의 구현으로 데이터의 흐름을 나타내는 것은 어렵다.

아이콘 데이터 흐름(Iconic Data Flow) 프로그래밍 환경은 이미지 프로세싱(image processing)이나 컴퓨터 비전에 대한 시스템을 개발하는 데 적합하게 구성되었다[1]. 데이터 흐름 언어는 언어의 특징상 프로그램 자체를 하나의 데이터 흐름도로 볼 수 있다. 따라서 데이터 흐름 언어로 구현된 프로그램은 목적 IDF 시스템의 구조에 상응시켜서 구현할 수 있다. 그렇기 때문에 이러한 IDF 프로그래밍 환경은 데이터 흐름 언어를 이용한 개발에 최적화 되어 구성되었다. 이 연구에서는 IDF 프로그래밍 환경을 제공하고는 있지만 설계에 대한 고려는 하고 있지 않다.

UML을 적용하여 내장형 시스템을 모델링 하는 연구가 있었다[5]. 내장형 시스템을 모델링 하는데 있어서는 프로세스 네트워크(Process Networks)나 제어/데이터 흐름도(Control/Data Flow Graphs)같은 데이터 흐름 지향적인 표기법이 많이 사용된다. 이런 점에서 내장형 소프트웨어의 모델링에 가장 효과적인 방법으로 데이터 흐름도를 꼽고 있다. UML로 데이터 흐름을 효과적으로 나타내기 위해 액티비티 다이어그램(activity diagram)과 교류도를 두 개의 수정 후보 다이어그램으로 제시하였다. 그리고 그 중 교류도를 선택하여 데이터 흐름도를 표현하는 방법을 제안했다. 그러나 UML 메타모델을 수정 또는 확장하였는지에 대한 방법과 이러한 확장된 교류도가 적용된 구체적인 사례가 나타나 있지 않다.

3. UML 확장 메커니즘

이 장에서는 데이터 흐름 언어를 위한 UML 확장 메커니즘을 설명한다. 먼저 향후의 데이터 흐름 언어로 다양한 응용 프로그램을 구현하는 것에 일관되게 적용하기 위해 UML 메타모델의 개요(profile)를 새롭게 정의 하고, 그 다음에 이 개요의 정의 원칙을 밝힌다.

데이터 흐름 언어는 다음과 같은 중요한 구성 요소를 갖는다.

- 데이터
- 데이터의 연산 단위
- 데이터의 흐름

이런 데이터 흐름 언어의 구성 요소는 객체 지향적이지 않기 때문에 기존의 UML로 이를 나타내는 것은 다소 무리가 있다. 따라서 기존의 UML을 확장하여 데이터 흐름도 등을 표현하는 대안적인 표기법이 요구된다.

본 연구에서는 UML의 클래스 다이어그램으로 시스템의 구조를, 액티비티 다이어그램으로 시스템의 데이터 흐름을 표현한다. 먼저 데이터 처리 하위 단위를 UML의 확장된 클래스로 나타내고 각 데이터의 연산 단위간의 포함 또는 사용 관계를 UML의 클래스 다이어그램으로 나타낸다. 이후에 각각의 확장된 클래스간의 데이터의 흐름은 액티비티 다이어그램에서 나타낸다. 데이터 흐름 언어에서 나타나는 특이한 형태의 포대기형(nested) 데이터는 UML 액티비티 다이어그램의 포대기형 상태를 사용하여 표현한다. 데이터 흐름 언어를 위해 기존의 UML 메타모델을 확장하여 새로 정의한 UML의 메타모델 개요는 그림 1에서 보여 진다.

그림 1에 나타난 확장된 메타모델은 실제 모델링에 응용된다. 개요에 나타난 스테레오타입 <<Data>>는 데이터 흐름 언어의 데이터를 확장된 상태로 나타낸다. 스테레오타입 <<Process>>은 시스템의 데이터 처리의 기능적 단위(functional unit)를 액티비티 다이어그램의 확장된 액티비티와 클래스 다이어그램의 확장된 클래스로의 중복 상속을 통해 각각 데이터 처리의 동적인 면과 정적인 면을 나타낸다. 마지막으로 스테레오타입 <<Call>>은 클래스 다이어그램에서 스테레오타입 <<Process>>의 확장된 클래스간의 사용 관계를 확장된 의존 관계(dependency)로 나타낸다.

중요한 UML 확장 원칙을 요약하면 다음과 같다.

- 모든 것은 오직 UML의 구성물(entity)로만 표현한다.
- 데이터 흐름 언어가 객체와 클래스로 사상(mapping)

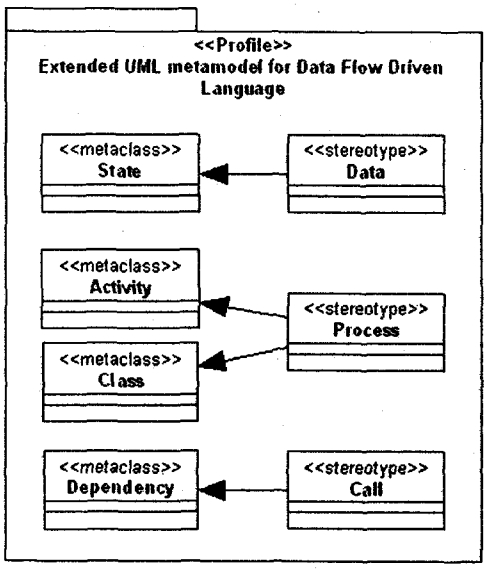

그림 1. 데이터 흐름 언어를 위해 확장된 UML 메타모델 개요

될 수 있는 특성내지는 구조를 가지고 있다면 그것을 UML의 다이어그램의 구성물로 표현한다. 사상된 UML의 구성물은 기존의 UML의 구성물과 구분하기 위해 새로이 스테레오타입을 정의해준다.

- 데이터 흐름 언어의 특성 또는 구조가 객체 지향 언어와 유사한 점이 거의 없다면 UML의 구성물의 조합을 통해 데이터 흐름을 나타낸다.
- 데이터 흐름 언어의 구성 요소를 캡슐화(encapsulation)를 할 수 있어야한다. 이러한 시스템의 표현은 관계자들이 시스템의 설계와 개발 산출 문서를 잘 이해할 수 있게 해준다.

4. 사례 연구

우리는 분산공유형 건설연구인프라 구축사업 추진연구[1] 프로젝트에서 MTS(Mini-shaker Tele-operation System)와 HST(Hybrid Shaking Table)라는 지진 모사 시스템을 설계, 개발하였다. 이 두 시스템은 다음과 같은 요구 사항을 만족해야한다.

- 각 시스템은 실험 시설을 원격지에서 사용자가 제어할 수 있어야 한다.
- 실험자가 실험에 필요한 데이터를 실험 시설 제어 프로그램에 보내어 실험을 수행하고 실험 결과를 데이터 저장소에 저장한다.
- 실험자는 실험 과정을 실시간으로 관찰할 수 있어야 한다.

MTS나 HTS에서 동작하는 실험 시설은 거대한 전체 그리드 시스템의 일부 실험 시설이다. 이런 복잡하고 거대한 시스템은 크게 3개의 하위 시스템, 즉 클라이언트 시스템, 서버 시스템, 기계 장치 제어 시스템으로 나뉘어서 개발 되었다. 이러한 큰 하위 시스템은 각기 이종적인 언어로 구현되었는데, 이는 한 가지 언어로 다른 성질의 하위 시스템들을 모두 개발한다는 것은 매우 비효율적이기 때문이다. 클라이언트 시스템 개발, 서버 시스템 개발, 기계 장치 제어 시스템의 개발에는 각각 MATLAB, Java, LabVIEW[7]가 사용되었다. MATLAB은 기능적 언어(functional language), Java는 객체 지향 언어, LabVIEW는 데이터 흐름 언어 중의 하나이다. 설계와 산출물의 일관성을 위해 우리는 표준 모델링 언어를 선택해야 했고, 이에 UML을 선택 하였다.

표 1. LabVIEW를 위한 UML 확장 표기법 - 스테레오타입

표기법	뜻
<<Process>>	클래스 다이어그램과 액티비티 다이어그램에서 기능적 단위를 나타내는 스테레오타입으로 VI 등의 데이터 처리 구조로 구현된다.
<<Data>>	액티비티 다이어그램에서 기능적 단위들이 처리하는 데이터를 나타내는 스테레오타입이다. 데이터는 기능적 단위간의 와이어를 통해 전달된다.
<<Call>>	클래스 다이어그램에서 기능적 교류도 간의 사용관계를 나타내는 스테레오 타입이다. VI가 다른 VI 등의 기능적 단위를 사용하는 경우 이를 표현한다.

여기서 하드웨어 제어 하위 시스템의 구현 언어인 LabVIEW는 National Instrument 사에서 개발한 데이터 흐름 언어로 가장 유명한 데이터 흐름 언어 중에 하나이다. LabVIEW는 실험 기기 제어 프로그램의 구현에 널리 쓰이고 있다. LabVIEW는 기능적 단위인 VI(Virtual Instrument)를 통해 데이터를 처리 한다. VI는 하위 VI를 포함할 수 있고, 데이터 역시 다른 데이터를 포함할 수 있는 구조로 되어있다. 이러한 데이터는 VI로 들어가서 처리되고 결과 데이터는 다음 VI로 넘어가는 흐름을 가진다. VI간에는 와이어(wire)라는 데이터의 연결통로로 이어져 있어서 데이터는 이 와이어를 따라 다음 VI로 전달된다. 기존의 UML의 표현법으로는 이러한 LabVIEW의 데이터 흐름에 대한 특징을 표현할 수가 없다. 우리는 앞서 정의한 UML 메타모델 개요에 따라 확장된 UML의

1) http://www.koced.net

표 2. LabVIEW를 위한 UML 확장 표기법 - 데이터 흐름도 구성 요소

구성 요소	표기법	설 명
프로세스	<<Process>> FeedbackCalculation	<<Data>>를 입력으로 받아서 그 데이터를 처리하여 어떠한 연산이나 기계의 컨트롤을 하는 기능적 단위를 나타내는 구성 요소이다. 그 결과로 인한 <<Data>>가 있으면 내보낸다.
데이터	<<Data>> Wave Data	<<Process>>의 입출력에 해당하는 데이터를 나타내는 구성 요소이다. <<Process>>간에 연결되어있는 와이어를 통해 전달된다. 조건 분기정의 조건 데이터의 역할도 한다.
포대기형 데이터	<<Data>> Base Data <<Data>> Base Displacement <<Data>> Base Velocity	LabVIEW가 가지고 있는 특별한 데이터 형태로 데이터간의 포함관계를 나타내는 구성 요소이다. 포함되어있는 '내부 데이터'들은 각각 독립적으로 <<Process>>나 조건 분기정의 입력으로 들어갈 수 있다. 데이터가 여러 데이터를 모아서 하나의 의미를 나타내는 경우 이러한 포함관계를 나타내는데 사용된다.
전역 데이터	Global	LabVIEW에서 사용되는 전역 데이터를 나타내는 구성 요소이다. 전역 데이터는 전역 VI에 따로 저장되어서 사용되는 구조를 갖는다.
조건 분기점	No No Input Data?? Yes	데이터의 컨트롤 흐름에 분기가 있을 경우 분기 경로와 분기 조건을 나타내는 조건 분기점이다. <<Data>> 상태나 <<Process>>의 연산 결과에 따른 분기가 있을 때 다음 제어 흐름을 표시해주는 분기를 나타내는 구성 요소이다.

두 개의 다이어그램, 클래스 다이어그램과 액티비티 다이어그램을 사용하여 LabVIEW의 특성들을 표현하였다.

표 1은 LabVIEW로 구현되는 시스템의 모델링을 위해 확장된 UML의 표기법 중 스테레오 타입에 대한 설명을 하고 있다. 표 2는 LabVIEW의 동적인 면을 모델링 하는 액티비티 다이어그램의 표기법에 대한 설명을 하고 있다.

이러한 표기법에 의해 설계한 HST는 다음과 같은 요구 사항을 충족하도록 개발했다.

- 입력 데이터는 로컬 머신(local machine)에 저장되어 있는 파일로 전달한다.
- 입력 데이터인 지진파 정보를 가지고 진동대의 움직

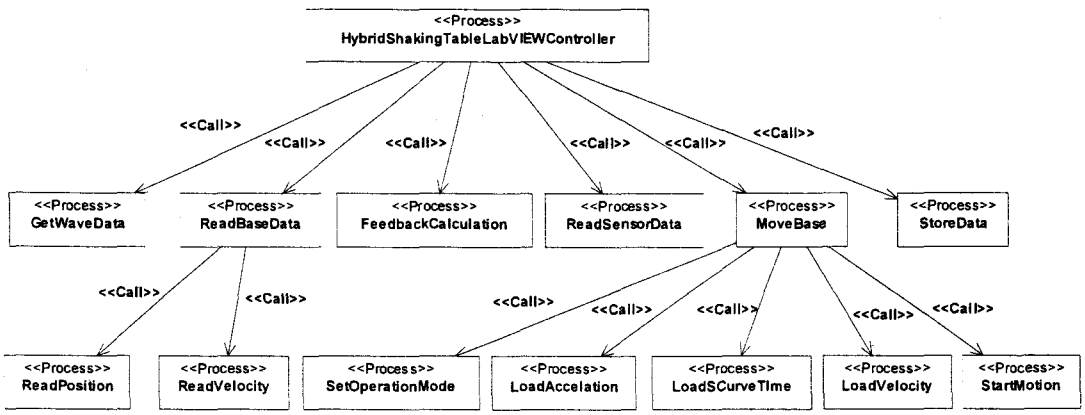

그림 2. HST의 정적 모델링 - 확장된 UML 클래스 다이어그램

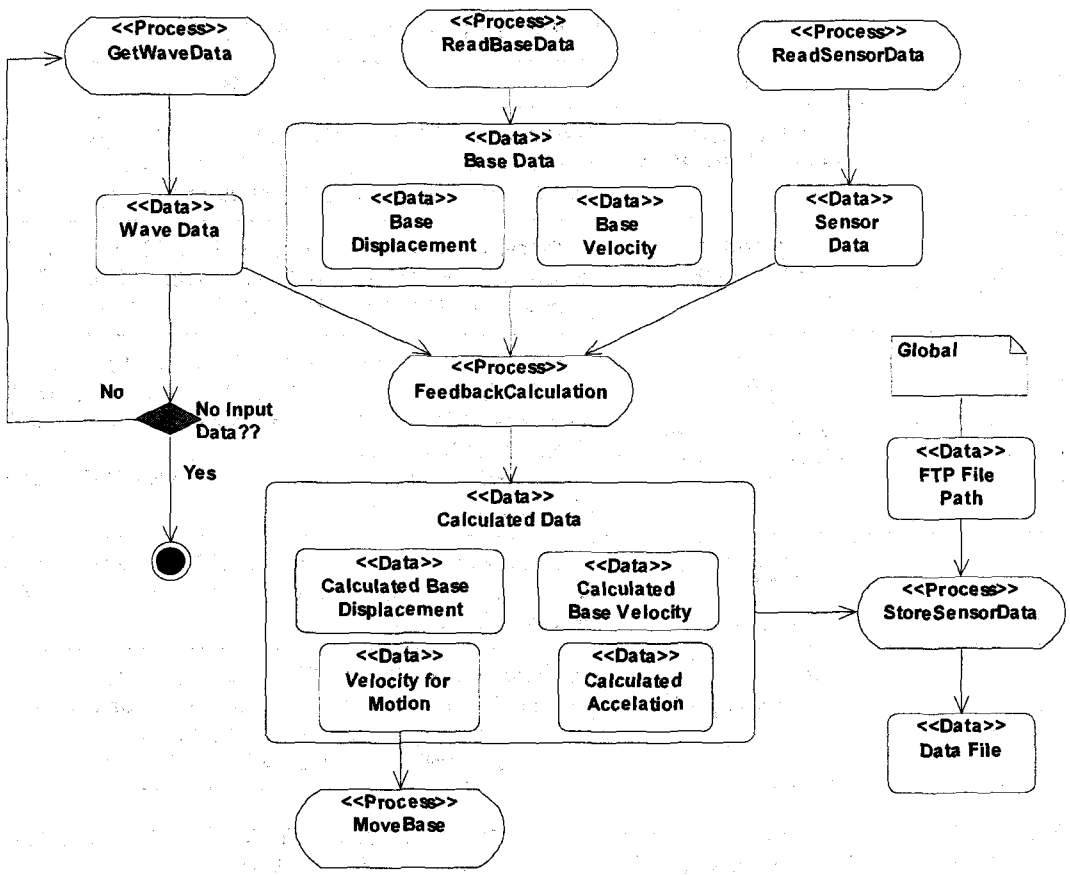

그림 3. HST의 데이터 흐름도 - 확장된 UML 액티비티 다이어그램

임을 계산하여 진동대를 구동한다.

- 센서를 통해 구동된 진동대의 움직임에 대한 데이터를 얻어서 그 데이터는 다음 지진파 데이터와 함께 그 다음 진동대의 움직임을 계산하는데 사용한다.
- 진동대의 움직임에 대한 데이터는 파일 전송 통신규약(File Transfer Protocol)를 통해 파일로 저장한다.
- 입력 파일로부터 더 이상의 지진파 데이터가 들어오지 않으면 프로그램은 종료한다.

그림 2는 이러한 명세와 실제 실험 시설의 정보에 따라 기능적 교류도에 대한 정적 모델링을 확장된 클래스 다이어그램으로 나타낸 것이다. 그림 2에서 <<Process>> *ReadBaseData*는 기계 장치에 대한 기본 데이터를 얻기 위해 <<Process>> *ReadPositon*과 <<Process>> *ReadVelocity*를 사용한다. 이러한 기능적 단위간의 사용 관계가 <<Call>>로 나타난다.

그림 3은 기능적 단위에 대한 정적 모델링을 바탕으로 실제 데이터 흐름도를 확장된 액티비티 다이어그램으로 나타낸 것이다. 실제 기계가 작동하는데 필요한 정보들을 바탕으로 각 기능적 단위간의 데이터 전달과 기능적 단위가 처리하여 내보내는 데이터에 대한 모델링을 한다. 포대기형 데이터인 <<Data>> *Calculated Data*는 4개의 <<Data>>를 포함하고 있다. 이 포대기형 데이터는 그 자체가 <<Process>> *FeedbackCalculation*의 결과물로 나온 계산된 데이터를 뜻하고, 이 데이터는 <<Process>> *StoreSensorData*의 입력으로 들어가 FTP를 통해 파일로 저장되게 된다. 하지만 <<Data>> *Calculated Data*에 포함된 <<Data>>중 <<Data>> *Velocity for Motion*은 기계를 구동하는 <<Process>> *MoveBase*의 입력으로 사용되어 기계를 실제 구동하게 된다. 이렇게 LabVIEW에서 이러한 포대기형 데이터는 내부 데이터를 필요에 따라 유동적으로 뽑아서 쓸 수 있고, 새로운 데이터를 포함시켜줄 수 도 있다. <<Data>> *FTP File Path*는 이후 원격 제어가 가능해지도록 다른 모듈과 연결될 때 다른 모듈도 사용할 수 있게끔 전역

데이터로 지정하였다. 그리고 <<Process>> *GetWaveData*의 결과로 더 이상 지진파 데이터가 없을 경우에 프로그램을 종료시키기 위해 조건 분기점을 달아서 입력 지진파 데이터가 있는지 없는지를 검사를 한다.

이 확장된 UML을 통해 얻은 HST의 데이터 흐름도를 통해 우리는 성공적으로 LabVIEW 프로그램을 개발할 수 있었다. 데이터 흐름도에 나타나 있는 <<Process>>와 <<Data>>를 가지고 중요 VI와 거기에 필요한 데이터, VI 간의 와이어링(wiring)을 하였다. 그 후 중요 VI가 필요로 하는 세부 VI를 확장된 클래스 다이어그램을 참고하면서 구현하였다. 이러한 과정은 매우 성공적이었으나, 구현 이전에 확장된 클래스 다이어그램에서 데이터 흐름도를 뽑아내는 과정은 매끄럽지 못했다. 이는 각각의 <<Process>>가 필요한 데이터가 구체적으로 어떤 것인지 확장된 클래스 다이어그램에 나타나 있지 않기 때문이다. 그렇기 때문에 클래스 다이어그램에 해당 <<Process>>의 입력 데이터와 출력 데이터를 표기해주는 방법이 필요하다.

5. 결론

지금까지 객체 지향 언어가 아닌 데이터 흐름 언어를 UML로 모델링하기 위한 UML 확장 메커니즘에 대해 설명했다. 데이터 흐름 언어에서 데이터와 데이터를 처리하는 기능적 단위들을 클래스나 상태 등으로 사상하는 메타모델 개요를 정의하였다. 그리고 이러한 메타모델 개요를 데이터 흐름 언어인 LabVIEW에 적용하였다. LabVIEW를 위해 확장된 클래스 다이어그램과 액티비티 다이어그램을 이용하여 HST를 설계하여 개발하였다.

본 연구에서 제시한 UML 확장 메커니즘을 응용하여 다른 비 객체 지향 언어들을 위해 UML을 확장할 수도 있을 것이다. 다양한 이종적인 언어들에 대한 UML 확장을 계속해나가면 UML을 통한 복잡하고 거대한 시스템의 통합 설계와 문서화에 도움이 될 것이다.

본 연구에서는 일반적인 데이터 흐름 언어의 특징을 표현하는데 중점을 두고 UML 메타모델 개요 정의를 했기 때문에 특수한 데이터 흐름 언어에서는 표기법이 부족할 수도 있다. 또한 확장된 클래스 다이어그램에서 데이터 흐름도를 도출하는 과정을 명확하게 하기 위해서 클래스 다이어그램에 각각의 데이터 연산 단위의 입출력 데이터를 표기하는 방법이 필요하다. 나아가 다양하고 독특한 데이터 흐름 언어에 대한 지속적인 연구를 통해 확고한 UML 메타모델 개요를 정의할 것이다. 그리고 다양한 이종적인 언어로 구현된 시스템을 위해 다양한 언어에 대한 지속적인 UML 확장을 할 것이다. 이렇게 UML을 확장해 나가면서 일반화 시킨다면 궁극적으로는 각각의 언어에 맞는 UML의 메타모델 개요에서 표준화된 전체 UML 메타모델을 얻을 수 있을 것이다.

참고문헌

[1] Hunt, N., "IDF: A graphical data flow programming language for image processing and computer vision", *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 351 - 360, 1990.

[2] Bin Jusoh, R., Ibrahim, M., and Mohd Nor Berhan, "Design, develop and test software to study the rate of climb for GPS paraglide", *Proc. of the Asian Conference on Sensors and the International Conference on new Techniques in Pharmaceutical and Biomedical Research*, pp. 168 - 172, 2005.

[3] Sommerville, I., *Software Engineering. Eighth Edition*, Addison-Wesley, 2004.

[4] Rumbaugh, J., Jacobson, I., and Booch, G., *The Unified Modeling Language Reference Manual. Second Edition*, Addison-Wesley, 2005.

[5] Fernandes, J.M., and Lilius, J., "Functional and Object-Oriented Views in Embedded Software Modeling", *Proc. of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pp. 378-387, 2004.

[6] Chang, C.K., Cleland-Huang, J., Hua, S., and Kuntzmann-Combelles, A., "Function-Class Decomposition: A Hybrid Software Engineering Method", *IEEE Computer*, pp. 87-93, 2001.

[7] National Instruments Co., *LabVIEW User Manual. Second Edition*, 2000.