AWT 기반 어플리케이션을 이클립스 플러그-인으로 리엔지니어링 할 때 재사용을 위해 고려할 사항들†

(Reengineering guidelines to develop eclipse plug-ins using AWT based legacy application)

양진석‡ 손동렬‡ 강교철¶

(Jin-Seok YANG) (Dong Ryul Dong) (Kyo. C. Kang)

요 약 우리는 소프트웨어 제품라인 개발 방법론인 FORM을 지원하기 위해 과거 AWT기반의 자바 어플리케이션으로 개발된 ASADAL의 동작 속도의 개선, 기능 추가, 그리고 좀 더 친숙한 사용자 인터페이스로의 개선이 필요했기 때문에 이클립스 플랫폼 기반의 플러그-인 어플리케이션으로 리엔지니어링을 진행해야 했다. 본 논문에서는 새로운 플러그-인 어플리케이션을 최소의 노력으로 개발하고 좀 더 빨리 배포하기 위해서 기 개발된 어플리케이션의 많은 부분을 재사용 할 수 있도록 리엔지니어링 과정에서 고려해야 할 세 가지 요소인 기 개발된 어플리케이션의 사용자 인터페이스 재사용 범위, 어플리케이션을 구성할 플러그-인들의 구성, 그리고 이후의 손쉬운 확장 및 기능 대체를 위한 플러그-인들 사이의 결합에 대해서 언급하고 각 요소에 대해서 사용한 방법 및 지침들을 소개한다. 그리고 ASADAL의 휘처모델 편집기와 행위모델 편집기의 리엔지니어링 사례를 들어 제시한 지침을 적용하여 개발한 플러그-인을 소개한다.

키워드 이클립스, 플러그-인 개발, AWT, 리엔지니어링

Abstract In order to improve working speed, add more functions and provide better user interface of ASADAL, the AWT based Java application to support FORM Software Product Line Development Methodology; we had to reengineer this application to be Eclipse based plug-in application. In order to improve working speed, add more functions and provide better user interface of ASADAL, the AWT based Java application to support FORM Software Product Line Development Methodology; we had to reengineer this application to be Eclipse based plug-in application. And then, this paper introduces the plug-in developed by applying guidelines presented in the reengineering cases of ASADAL's feature model editor and behavior model editor.

Key words Eclipse, Plug-in Development, AWT, 리엔지니어링

† 본 논문은 정보통신산업 진흥원의 소프트웨어 공학 요소기술 연구개발 사업에 의해 지원되었음을 밝힙니다.
The research was supported by WCU (World Class University) program throuth the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-2008-000-10100-0).

‡ 비 회 원 : 포항공대 정보통신 연구소
edeward@gmail.com
‡ 비 회 원 : 포항공대 정보통신 연구소
mymy29@gmail.com
¶ 종신회원 : Devision of ITCE, POSTECH
kck@postech.ac.kr

논문접수 : 2011년 09월 05일
심사완료 : 2011년 09월 25일

1. 연구배경

포항공대 소프트웨어공학 연구실에서 개발한 ASADAL [1]은 과거 10년 동안 소프트웨어 제품라인 공학의 개발 방법인 FORM (Feature-Oriented Reuse Method) [2]을 지원하기 위한 도구로써 개발되어 왔다. 자바 (Java) 진영에서 개발된 그래픽 라이브러리 AWT/Swing 기반으로 개발된 ASADAL은 두 가지 문제점에 대한 리엔지니어링이 필요로 하고 있다. 첫 번째는 오늘날

AWT의 근본적인 문제로 지적되고 있는 친숙하지 못한 사용자 인터페이스 룩 앤 필 (Look and Feel) 문제와 메모리 누수 문제로 부터 발생하는 성능적 문제이다.[3] 두 번째 개발환경에서 부터 기인하는 것으로 기 개발된 컴포넌트와 독립적으로 기능적인 확장이 용이한 개발환경을 개발자에게 제공하는 것이다. 과거의 독립형 자바 어플리케이션 개발환경에서는 기 개발된 컴포넌트와 새롭게 개발되거나 확장된 컴포넌트의 구분과 관리가 용이하지 못하기 때문이다. 그 결과 다양한 제품이 발생하게 되고 제품들 사이의 기능 및 데이터 모델의 일관성을 유지하지 못하는 문제가 발생하기도 하였다.

우리는 위의 리엔지니어링을 통해 FORM 제품라인 방법론을 지원하는 새로운 도구를 최대한 빠른 시간에 사용자들에게 배포하고 싶었다. 그러기 위해서는 기 개발된 ASADAL을 구성하는 사용자 인터페이스를 포함한 주요 컴포넌트를 재사용해야 했다. 그리고 장기 개발 로드맵에 따라서 재사용된 컴포넌트들을 점진적으로 새로운 기능으로 손쉽게 대체 할 수 있는 개발환경이 필요했다. 그 결과 우리는 ASADAL 개발 당시에 사용된 자바 플랫폼 보다 기술적으로 많이 발전한 이클립스 RCP (Rich Client Platform)를 기반으로 리엔지니어링을 진행하여 이클립스 플러그-인에서 재사용 할 수 있도록 ASADAL 컴포넌트들을 플러그-인으로 이송 (Migration) 하여 새로 개발된 플러그-인들과 함께 사용하고 점차적으로 플랫폼으로 이송된 컴포넌트를 신규 컴포넌트로 대체하기로 결정하였다. IBM에서 개발을 주도하고 있는 이클립스 플랫폼은 자바 어플리케이션을 위해 SWT/Jface기반의 사용자 인터페이스, 작업공간, 로깅 등의 다양한 편의 기능을 제공하여 AWT기반 자바 어플리케이션에서 지적되던 사용자 인터페이스와 성능 문제를 해결하고, 기능적 확장이 용이한 OSGi (Open Services Gateway Initiative) 플러그-인 아키텍처를 지원하고 있기 때문이다. NASA등의 기관에서 이클립스 플랫폼 기반의 어플리케이션을 개발하여 사용하고 있어 어플리케이션 개발 편의성과 안정성을 대변하고 있다. [4]

본 논문에서는 위와 같은 목적을 달성하기 위해 기 개발된 응용소프트웨어의 재사용 부분을 플러그-인으로 이송하는 내용을 포함한 리엔지니어링에서 고려할 사항과 이에 대한 개발방법 및 지침에 대해서 설명하고 있다. 2장에서는 과거의 플랫폼과 신규 플랫폼에 대해서 간략하게 비교하고, 리엔지니어링 목표에 대해서 설명한다. 3장에서는 2장에서 정의한 목표를 달성하기 위한 리엔지니어링에서 고려해야 할 요소와 지침에 대해서 설명한다. 그리고 리엔니지어링 과정을 통해 실제 개발된 ASADAL의 휘처 모델링 지원도구와 행위모델링 지원도구 플러그-인에 대해서 설명한다. 마지막 4장에서는 결론과 향후계획을 설명한다.

2. 연구배경

이 장에서는 과거 ASADAL의 운영 플랫폼과 새로운 개발 도구의 플랫폼 사이의 비교와 새로운 어플리케이션으로 이송될 ASADAL의 기능에 대해서 간략하게 설명한다.

아래의 표 1은 두 자바 어플리케이션 플랫폼들의 차이점을 간략하게 보여주고 있다. 플러그-인 어플리케이션들은 어플리케이션 본연의 기능들 이외의 로깅 (Logging), 그래픽 라이브러리, 그리고 문서 편집 등을 위한 기능을GEF나 EMF등을 사용함으로써 별도의 개발 없이 사용 할 수 있는 장점을 가지고 있다. 과거 어플리케이션에서 이러한 기본 기능을 직접 개발하던 것을 이클립스 플랫폼을 사용함으로서 해당 기능의 개발시간을 절약 할 수 있다.

표 1. 개발 플랫폼들의 비교

Spec	As-Is	To-Be
Application Type	Standalone	Plug-ins
JVM	1.4	1.5+
Graphic Library	AWT/Swing	SWT/JFace
Model 관리	직접개발	EMF
Framework	직접개발 (Gdt 등)	Eclipse Platform (Command, EMF,GEF,Log등)

우리는 ASADAL의 많은 기능들 가운데 휘처 모델링 지원도구와 행위 모델링 지원도구를 우선 플러그-인으로 새롭게 개발하기로 했다. 그리고 새로운 기능을 추가적으로 반영하여 개발하기로 했다.

FORM에서 핵심이 되는 휘처 모델링 지원도구는 기존 도구에 비해 아래와 같은 차이점을 가지고 있다.

FORM보고서에 명시된 휘처 모델링 요소의 완전한 호환성 및 추가 모델링 요소들의 반영에 따른 데이터 모델 구조

다른 도구들과의 호환을 위한 XML형식의 데이터 저장

다이어그램 이외의 휘처 모델의 트리 및 그래프를 이용한 다양한 뷰 (View) 형태 제공

행위 모델링 지원도구는 Statechart, DFD (Data Flow Diagram), 그리고 다이어그램에서 사용되는 이벤트 또는 데이터의 정의하기 위한 어트리뷰트 편집기로 구성이 되어 있다. 다이어그램 기반의 편집기능이 대부분을 차지하는 이 도구에서 휘처 모델의 경우와 같이 데이터 모델의 변경은 하지 않고 메뉴나 어트리뷰트 편집기에서의 사용자 인터페이스만을 새롭게 수정하기로 했다. 왜냐하면 휘처 모델 도구와 달리 다이어그램 기반의 편집기들은 신규 개발 시 많은 신간을 소요하기 때문이다.

3. 리엔지니어링에서 고려할 사항들

이 장에서는 우리는 기 개발된 어플리케이션을 최대한 재사용하여 앞 장에서 설명한 두 도구들을 플러그-인으로 새롭게 개발하기 위해 고려한 사항들과 개발을 위해서 마련한 지침에 대해서 소개한다.

리엔지니어링은 AWT기반 어플리케이션의 사용자 인터페이스의 재사용 범위 결정, 패키지 분석과 플러그-인 설계 그리고 신규 플러그-인과 이송된 플러그-인들 사이의 연결 순서로 진행이 되었다.

3.1. 사용자 인터페이스 재사용 범위 결정하기

현재 대부분의 윈도우 어플리케이션의 사용자 인터페이스는 고착화 되었다고 할 만큼 유사한 모양의 컴포지트 (Composite) 와 위젯 (Widget)으로 구성되어 있다. 그래서 기본 윈도우 인터페이스를 굳이 과거의 사용자 인터페이스 재사용할 필요는 없었다. 하지만 SWT 기반의 이클립스 플랫폼의 윈도우 사용자 인터페이스에서는 제공하지 않는 개념인 MDI (Multi Document Interface)와 일반적인 윈도우 어플리케이션 위젯으로는 구현할 수 없는 다이어그램 편집기에 대해서는 플러그-인 개발에 어떻게 반영을 할지 결정이 필요했다.

MDI 개념의 AWT 내부 프레임 (Internal Frame)을 ASADAL에서는 목적에 따라 여러 개의 기능들을 하나의 프레임으로 묶고 작업공간을 나누기 위한 용도로 사용을 하고 있었기 때문에 플러그-인으로 이송시 반드시 가져가야 할 요소는 아니었다. 그래서 이 역할을 이클립스 플랫폼의 뷰 파트 (View Part)와 에디터 파트 (Editor Part)로 대체하기로 했다. 이 두 요소는 어플리케이션을 구성하는 플러그-인들의 여러 기능을 작업공간상에 배치하는 것을 목적으로 설계된 이클립스의 요소들 이기 때문에 AWT 기반 어플리케이션의 MDI의 훌륭한 대안이 될 수 있다.

신규 플러그-인의 빠른 배포를 위해서는 개발에 많은 시간과 노력을 필요로 하는 다이어그램은 재활용 대상으로 아주 매력적인 요소이다. 이 매력적인 요소가 초기 배포에 도움을 주면서 이후 이클립스 플랫폼의 기본 사용자 인터페이스에 좀 더 적합한 GEF (Graphical Edit Framework) 또는 GMF (Graphical Modeling Framework) 기반의 새로운 다이어그램 편집기로 손쉽게 대체 될 수 있게 플러그-인으로 이송 될 수 있도록 해야 했다. 하지만 일반적으로 기 개발된 응용소프트웨어에서 다이어그램에 해당하는 편집 환경과 다이어그램 데이터 모델 등을 다른 모듈로 부터 정확하게 분리하여 나누는 것이 쉬운 작업이 아니기 때문에 우리는 재사용할 다이어그램을 위한 아래의 작업 지침들을 마련했다.

- 개발지침 1) 다이어그램을 위해서 필요한 기 개발된 응용소프트웨어의 패키지들을 모두 하나의 플러그-인으로 이송을 시킨다.
- 개발지침 2) 개발지침1에서 이송된 플러그-인에 신규로 개발할 플러그-인을 위한 인터페이스를 정의한다.
- 개발지침 3) 신규 개발 플러그-인에서는 이송된 플러그-인에서 제공하는 인터페이스를 통해서 다이어그램 사용자 인터페이스 인스턴스를 반환받고 받환 받은 인터페이스를 SWT-AWT 브릿지 기술을 [5]이용하여 SWT 안에 포함 (Embedded) 시킨다.

휘처 모델링 지원도구와 행위 모델링 지원 도구에서 재사용 하기로 결정한 사용자 인터페이스는 아래의 그림 1과 그림 2에서 붉은 사각형 영역의 다이어그램 부분이다. 그 외의 사용자 인터페이스는 새로 개발될 플러그-인에서 새롭게 개발된다.

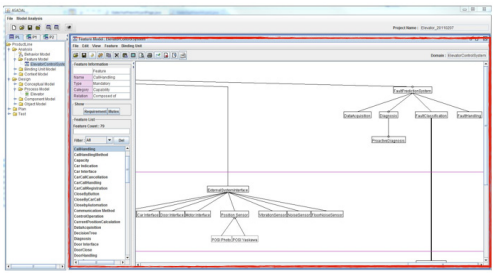

그림 1. ASADAL휘처모델링 도구 UI 재사용범위

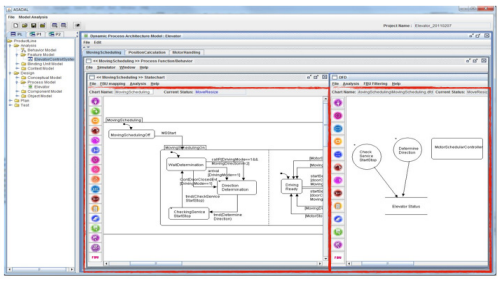

그림 2. ASADAL 행위모델링 도구 UI 재사용범위

3.2 패키지 분석과 플러그-인 구성

사용자 인터페이스에 대한 재사용 방향이 결정되었다면 어떻게 사용자 인터페이스를 포함하여 기 개발된 어플리케이션들의 패키지들을 플러그-인으로 이송하고 이송된 플러그-인들과 새로 개발해야 하는 플러그-인들 사이의 의존성 관계 구성에 대해서 고려해야 한다. 왜냐하면 이클립스 플랫폼 기반의 어플리케이션은 여러 개의 플러그-인들의 집합으로 구성되는 것이 일반적이고, 플러그-인들 사이의 의존성 관계가 복잡하면 어플리케이션을 구성하는 플러그-인을 대체하거나 새로운 플러그-인을 통한 기능 확장이 쉽지 않기 때문이다. 또한 AWT 기반의 어플리케이션 보다 빠른 응답성을 보장하기 위해서 이클립스 플랫폼에 제공하는 플러그-인의 늦은 바인딩 (Lazy Binding) [4] 기능을 십분 활용하기 위해서는 플러그-인들 사이의 의존성 관계 구성은 플러그-인 기반 어플리케이션 개발에서 상당히 중요한 문제이다.

우리는 어플리케이션을 구성하는 플러그-인들을 발굴하고 그들 사이의 의존성 관계를 구성하기 위한 아래의 개발 지침들을 마련하였다.

- 개발지침 4) 라이브러리 유형의 패키지는 독립적인 플러그-인으로 구성한다. 독립적인 플러그-인으로 구성함으로써 우리는 기능적 확장 측면에서 두 가지 장점을 얻을 수 있다. 첫 번째는 JAR형태로 라이브러리만을 별도로 배포할 수 있다는 점이다. 라이브러러 설정을 통해 개발자들은 관심이 없거나 작업 대상이 아닌 프로젝트를 작업공간에서 제거함으로써 작업 공간을 간결하게 유지할 수 있다. 두 번째는 차후 이 라이브러리만을 필요로 하는 새로운 플러그-인을 다른 플러그-인들과는 무관하게 개발 할 수 있다는 점이다.
- 개발지침 5) 휘처 모델링과 같이 대표 기능으로 분류될 수 있는 각각의 기능들은 패키지들 가운데 그 기능만을 담당하는 패키지들만 모아서 하나의 플러그-인으로 구성한다. 기존 패키지들을 최대한 구분하여 독립된 플러그-인으로 분리시킴으로써 패키지들 사이에 존재할 수 있는 다양한 의존관계를 정리 하고 이송 과정에서 인터페이스를 재정리 할 수 있는 장점을 가진다.
- 개발지침 6) 새롭게 개발할 플러그-인에는 기 개발된 어플리케이션의 패키지를 포함시키지 않는다. 신규 플러그-인과 기 개발된 어플리케이션에서 이송된 패키지를 포함하고 있는 플러그-인을 분리함으로써 차후 장기 개발 계획에 따라 기 개발된 어플리케이션에서 제공하는 기능의 대체를 용이하게 할 수 있다. 예를 들어 빠른 배포를 위해서 새로운 플러그-인에서 기 개발된 다이어그램 편집 환경을 사용하다가 새로운 다이어그램 편집환경의 사용 준비가 된 경우 기 개발된 다이어그램 편집환경의 기능을 제공하는 플러그-인의 의존관계를 끊음으로서 손쉽게 기능을 대체 할 수 있다.
- 개발지침 7) 앞에서 제시한 지침들에 따라 플러그-인을 발굴하고 의존 관계를 명시한 결과 플러그-인들 사이의 회전 의존성 (Cyclic Dependency) 이 발생하는 경우 회전 의존성을 야기하는 패키지 또는 클래스들을 독립된 플러그-인으로 구성하여 회전 의존성을 제거해야 한다. 이는 고품질의 이클립스 플러그-인 개발의 기본원칙 가운데 하나이다.

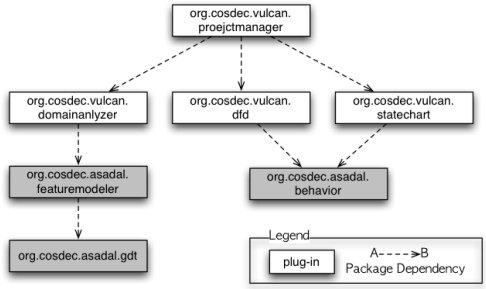

그림 3. 새로운 도구의 플러그-인들의 의존성 관계

우리는 위의 지침들을 바탕으로 새로운 어플리케이션의 플러그-인들을 위의 그림 3과 같이 구성을 할 수 있었다. 모두 7개의 플러그-인들 가운데 배경이 회색으로 표시된 3개의 플러그-인들은 재사용 범위에 따라 필요한 ASADAL의 패키지들을 이송한 플러그-인들이다. 흰색 바탕의 플러그-인들은 신규로 개발되는 플러그-인들이다. 이 가운데 "org.cosdec.vulcan.domainanalyzer" 는 휘처 모델링을 위한 새로운 사용 환경을 사용자에게 제공하고 휘처 모델의 뷰어들 가운데 다이어그램 뷰를 제공하기 위해서 ASADAL의 휘처 모델링 도구에서 제공하는 다이어그램을 사용한다. "*.vulcan.dfd"과 "*.vulcan.statechart" 플러그-인은 ASADAL의 Statechart 다이어그램과 Data Flow Diagram 편집 환경을 그대로 포함 (Embedding) 하고 있는 새로운 사용자 인터페이스를 사용자에게 제공한다.

3.3 이송된 플러그-인의 연결

어플리케이션을 구성하는 플러그-인들 사이의 관계가 정의가 되었다면 새로 개발될 플러그-인에서 이송된 플러그-인에 포함된 패키지의 내용의 수정을 최소화하여 연결하여 사용할 수 있는 방법에 대해서 고려해야 한다.

본 논문에서는 이송된 플러그-인을 사용하는 두 가지 경우에 대해서 다룬다. 첫 번째는 휘처 모델링 도구와 같이 새로 개발한 플러그-인에서 데이터 모델을 포함한 모든 부분을 새롭게 개발하고 이송된 플러그-인에서의 다이어그램 뷰(View) 만을 사용하기 원할 경우이다. 두 번째는 Statechart나 DFD의 경우와 같이 메뉴 등의 사용자 인터페이스 일부를 제외하고는 새로 개발된 플러그-인에서 이송된 플러그-인의 모델 및 뷰(View) 등의 대부분의 기능을 사용할 경우이다. 두 경우 모두 문제를 해결하기 위해서 적용된 소프트웨어 공학 기본 원칙은 정보은닉 (Information Hiding) 이다. [6]

첫 번째 경우 JFace의 프로바이더 (Provider) 개념을 차용하여 두 플러그-인들이 각각 사용하는 모델들 사이의 호환을 위한 모델 프로바이더를 제안하였다. (그림 4 참조) 모델 프로바이더는 신규 플러그-인의 모델을 이송된 플러그-인의 모델로 변환하여 이송된 플러그-인의 뷰에 모델을 공급하는 역할을 수행함으로써 이송된 플러그-인을 뷰로써 사용할 수 있게 만들었다. 이를 통해 차후 뷰어가 대체될 경우에도 손쉽게 대응 할 수 있게 되었다.

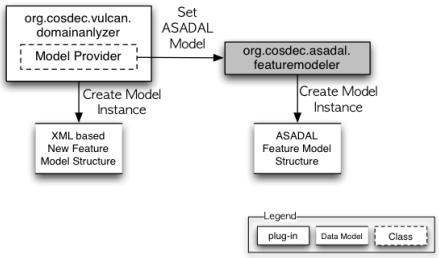

그림 4. 모델 프로바이더를 이용한 연결

아래의 [그림 5]와 [그림6]은 새로 개발된 휘처 모델링 지원도구가 모델 프로바이더를 통해 ASADAL의 휘처 다이어그램 뷰어를 SWT 사용자 인터페이스에 포함 (Embedding) 시켜 사용하고 있는 모습을 보여주고 있다.

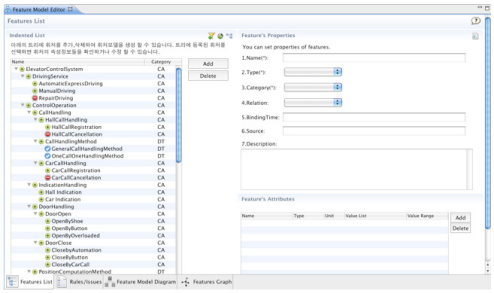

그림 6. 새로 개발된 사용자 인터페이스

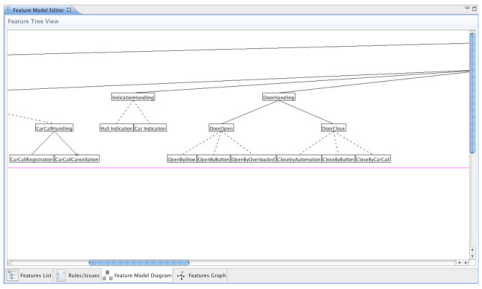

그림 7. 포함된 ASADAL 인터페이스

두 번째 경우를 위해서 이송된 플러그-인에서 제공하는 기능을 신규로 개발된 플러그-인에서 사용할 수 있는 어답터를 제공하였다. 이송된 플러그-인에서 제공하는 인터페이스를 직접 사용하지 않고 인터페이스로 동작하는 별도의 어답터를 이용하여 이송된 플러그-인으로 접근하게 함으로써 모델의 저장을 위해 미리 제공되는 이클립스 플랫폼의 사용자 인터페이스와의 연결을 위해서 이송된 플러그-인을 직접 수정하지 않고 개발을 진행 할 수 있었다.

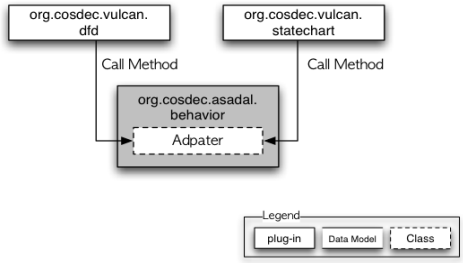

그림 8. 아답터를 이용한 플러그-인들 사이의 연결

아래의 그림 8과 그림 9는 기존 내부 프레임 사용자 인터페이스를 대체하도록 편집기 파트(Editor Part) 에 멀티 페이지 편집기로 구현된 Statechart와 Data Flow Diagram의 결과를 보여주고 있다. 이 두 개의 플러그-인들은 이클립스 플랫폼이라는 틀에서 사용자가 이송된 플러그-인의 기능을 사용할 수 있게 하는 통로로써의 역할만 수행한다. 이를 통해 사용자는 최소한의 이질감을 가지고 기 개발된 어플리케이션을 사용할 수 있게 된다.

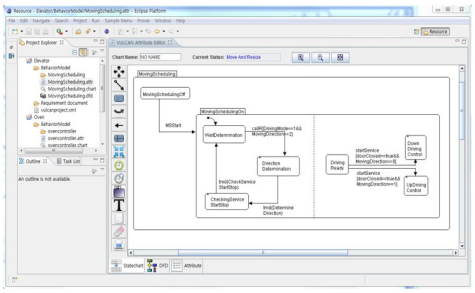

그림 9. 이송된 Statechart 편집기

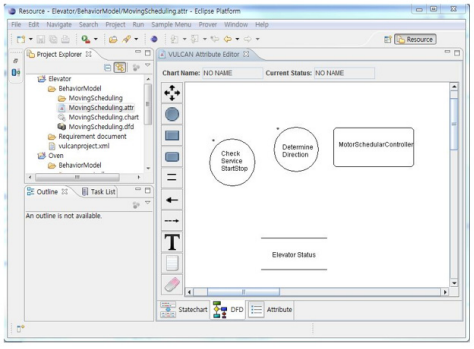

그림 10. 이송된 DFD 편집기

4. 결 론

본 논문에서는 신규로 개발된 플러그-인의 빠른 배포를 위해 재사용할 범위에 따라서 ASADAL의 일부 패키지를 플러그-인으로 이송하기 위한 지침들을 제시하고 활용 목적에 따라 이송된 플러그-인과 신규 플러그-인 사이를 모델 프로바이더와 아답터를 이용하여 플러그-인들 사이의 관계를 정의하고 연결하는 방법에 대해서 설명하였다.

휘처 모델링 지원도구와 행위 모델링 지원도구를 플러그-인으로 개발하는 사례를 통해 새로운 기능과 과거의 ASADAL 패키지를 활용하여 기 개발된 다이어그램을 연결하는 작업에 플러그-인당 1명의 개발자가 투여되어 약 1개월만 에 개발이 완료되는 것을 확인 할 수 있었다. 과거의 경험에 비춰 이클립스의 GEF와 EMF에 숙련된 플러그-인 개발자가 기본적인 다이어그램 편집 환경만을 개발할 때 1개월에서 2개월 정도의 개발시간을 필요로 하는데 비해 이 방법을 통해 훨씬 빠른 시간 안에 개발하고 배포를 할 수 있다는 것을 확인 할 수 있었다. 하지만 사용자 인터페이스 연결을 위해서 사용한 SWT-AWT 브릿지 기술이 어플리케이션의 운영 성능을 저하시키는 원인으로 작용하고 있어 목표로 삼은 어플리케이션의 성능 향상은 여전히 문제로 남아 있게 되었다. 그렇지만 이송된 플러그-인과 새로 개발된 플러그-인 사이를 모델 프로바이더와 아답터를 이용하여 느슨하게 연결함으로써 향후 성능 문제를 해결한 플러그-인으로 손쉽게 대체할 수 있게 개발을 함으로써 개발 계획에 따라 보완될 수 있을 것으로 예상된다.

우리는 본 논문에서 언급한 휘처 모델링 지원도구와 행위모델링 지원도구 이외의 ASADAL의 기능들을 점진적으로 이클립스 플러그-인으로 이송할 예정이며 개발에 참여하는 개발자가 이송을 일관되게 수행할 수 있도록 본 논문에서 제안한 개발 지침을 보완할 예정이다.

참 고 문 헌

[1] Kyungseok Kim, Jae Joon Lee, Miyoung Ahn, MinSeok Suh, Yeap Chang, Kyo Kang, ASADAL: A Tool System for Co-Development of Software and Test Environment Based on Product Line Engineering, The 28thInternational Conference on Software Engineering (ICSE2006), Shanghai, China, May 20-28, 2006, pp.780-786.

[2] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh, FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, 5, 1998, pp. 143-168

[3] Matthew Scarpino, Stephen Holder, Stanford Ng, Laurent Mihalkovic, "SWT/JFace In Action", Manning Publication, 2005

[4] Eric Clayberg, Dan Rubel, Eclipse: Building Commercial-Quality Plug-ins (2nd Edition), Addison-Wesley Professional, 2004

[5] Yannick Saillet, "ⓐMigrate your Swing application to SWT, Presented by developerWorks, your source for greate turorials", ibm.com/developer Works

[6] Hassan Gomma, "ÀDesigning Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures", Addison-Wesley, 2005

저 자 소 개

양 진 석

1999년 동국대학교 전산학과 졸업(학사)
2006년 고려대학교 컴퓨터학과 졸업(석사)
2006년~현재 포항공대 정보통신연구소 SE연구실 연구원.

<관심분야> 정형기법, 동기화 언어, 제품라인공학, 자동제어

손 동 렬

2003년 동국대학교 컴퓨터학과 (학사)
2009년~2010년 stx forcetec 근무
2010년~현재 포항공과대학교 정보통신연구소 연구원.

<관심분야> SW재사용, 객체지향 프로그래밍, 프레임워크

강 교 철

1973년 고려대학교 통계학과 졸업(학사)
1974년 콜로라도 대학교 산업공학 졸업(석사)
1982년~1984년 미시간대학교 Visiting Professor
1984년~1985년 (미국)벨 통신 연구소 Technical Staff
1985년~1987년 AT&T 벨 연구소 Technical Staff
1987년~1992년 카네기멜론 대학교 Project Leader
1992년~현재 포항공과대학교 교수
2000년~2001년 카네기멜론대학교 Visiting Scientist

〈관심분야〉 소프트웨어 공학, 실시간 내장형 시스템, 소프트웨어 재사용, 제품라인 공학