제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

# RAiSE: 다중 패러다임을 결합한 프로세스 모델링 언어

이형원, 최상일 0
강릉대학교 컴퓨터공학과
email : lhw@kangnung.ac.kr, cs9339[0]@cs.kangnung.ac.kr

# RAiSE: A Process Modeling Language Combining Multi-paradigm

Hyungwon Lee, Sangil Choi[0]
Dept. of Computer Science and Engineering, Kangnung National University

요 약

기존 프로세스 모델링 언어들은 엄격한 의미론과 사용의 용이성이라는 프로세스 언어가 갖추어야 할 중요한 두 가지 측면을 동시에 만족시키지 못함으로써 그 동안의 다양한 연구에도 불구하고 실제 소프트웨어 산업계에서 널리 사용되지 못하고 있다. 본 논문에서는 이러한 단점을 제거한 프로세스 모델링 언어 RAiSE 를 정의하고 이를 예제 소프트웨어 프로세스에 적용한 예를 설명한다. RAiSE 는 쉽게 이해할 수 있는 그래픽 표기법을 기반으로 하여 다양한 모델링 패러다임들의 필수적인 요소들을 결합시킴으로써 이해하기 쉬우면서도 엄격하고 풍부한 의미론을 제공한다.

## 1. 서론

프로세스 모델링 언어(PML)는 소프트웨어 프로세스 기술에 관한 관심이 모아지기 시작한 초기부터 지금까지 중요하게 강조되어온 연구 분야이다. 특히, PML 설계에 있어 크게 두 가지 상호 충돌되는 접근 방법이 사용되어 왔다[1,2]. 먼저, 프로세스는 조직, 행위, 산출물, 자원, 사건, 작업자, 예외 등의 다양한 의미론을 이용하여 기술될 수 있으며 또한, 그렇게 해야 한다는 주장에 입각해서 개발된 PML 들이 있다. 이러한 언어들은 강력하고 의미론적으로 매우 풍부하지만 텍스트 기반의 이해하기 어려운 형태로 표현해야 하며 매우 복잡하기 때문에 특히, 프로그래머가 아닌 사람들은 사용하기 어렵다. 반대로, 단순화에 초점을 맞춰 개발된 PML 들이 있는데 대개의 경우 의미론적 깊이와 너비를 제한하게 되며 그래픽 형태의 표현 방법을 사용한다. 이 언어들은 언어적 간결함을 강조하는 대신 실질적인 사용에는 제한을 받을 수밖에 없다. 즉, 대다수의 PSEE(Process-centered Software Engineering Environment) 에서 제공하는 PML 들은 엄격한 의미론과 사용의 용이성이라는 PML 이 갖추어야 할 중요한 두 가지 측면을 동시에 만족시키지 못한다. 효율적인 프로세스 실행과 분석을 위해서는 풍부하면서도 엄격한 의미론을 가져야 하며, 프로세스 모델링을 담당하는 대부분의 사람들이 프로그래밍 전문가가 아니며 프로세스 모델을 사용하는 기본 목적이 프로세스 이해라는 점에서는 쉽게 정의하고 이해할 수 있는 PML 이 제공되어야 한다.

본 논문에서는 상호 보완 관계에 있는 다양한 프로세스 모델링 패러다임을 하나의 모델 내에서 비전문가라도 쉽게 이해하고 정의할 수 있도록 표현해주는 그래픽 PML 인 RAiSE 를 소개한다. RAiSE 는 상태 기반의 행위 중심 모델링 패러다임을 근간으로 하여 주요 프로세스 모델링 패러다임의 핵심 개념들을 반영하는 그래픽 표기법을 제공한다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존 프로세스 모델링 패러다임들을 설명하고 3 장에서는 RAiSE 의 설계 목표를 기술한다. 4 장에서는 RAiSE 의 표기법을 정의하고 5 장에서는 RAiSE 를 ISPW-6 프로세스[3]에 적용한 예제를 설명한다. 마지막으로 6 장에서 결론 및 향후 연구 과제에 대해 논한다.

## 2. 기존 프로세스 모델링 패러다임

PML 은 프로세스의 다양한 측면을 표현하는데 유용한 구조들을 얼마나 지원하는가에 따라 평가될 수 있다. 소프트웨어를 모델링하고 프로그래밍하는데 사용되는 언어들 역시 프로그램이 수행되는 프로세스를

본 연구는 한국과학재단 목적기초연구(R05-2001-000-0105900) 지원으로 수행되었음

표현하고 실행시키는 수단이라는 점에 착안하여 대부분의 프로세스를 연구하는 사람들은 이러한 기존 언어들의 개념에 기초하여 PML 들을 제안하였다. 표 1 에는 프로세스 모델링 언어를 프로세스 모델링 패러다임에 따라 분류한 결과를 요약하였는데[4] 각각의 패러다임은 나름대로의 장단점을 갖고 있기 때문에 한 가지 패러다임만을 이용하는 것은 일관성 측면에서는 장점이 되지만 프로세스 모델링에 필요한 모든 요구 사항을 만족시킬 수 없다.

표 1 프로세스 모델링 패러다임

| 패러다임 | PML/PSEE |
|---|---|
| 비실행 | ETVX, IDEF0 |
| 상태 기반 | Statemate, SLANG, Process Weaver |
| 규칙 기반 | MARVEL, GRAPPLE, Adele, Merlin, OIKOS, EPOS, ELF |
| 프로그래밍 | APPL/A, Little-JIL |
| 정형적 언어 | Hakinowa, HFSP |
| 구조적 기법 | Statemate, IDEF0, PERT |
| 데이터 모델링 | AD model, PMDB, Statemate |
| 객체 지향 | EPOS, MARVEL, MVP, OBJ |
| 정량적 모델링 | System Dynamics |

## 3. RAiSE 의 설계 목표

RAiSE 의 설계 철학은 프로세스 모델은 프로세스의 각 스텝이 시작되고 종료될 때까지 어떠한 상태 변화를 일으키는가 그리고 상태 변화의 원인이 무엇인가를 명시적으로 표현해주어야 한다는 점에 바탕을 두고 있다. 이러한 상태 기반의 행위 모델링 패러다임을 기초로 하여 객체간 정보 전달을 나타내기 위하여 자료 흐름도 원리를, 스텝간 다양한 제어 흐름을 표현하기 위하여 프로세스 프로그래밍 개념을, 개체 상속을 표현하기 위하여 객체 지향 개념을 도입하였다. 기타 RAiSE 의 설계 목표는 다음과 같다.

**사용의 용이성**: 소프트웨어 엔지니어라면 누구라도 쉽게 이해하고 사용할 수 있는 문법과 의미론을 제공한다.

**프로세스의 계층적 분할**: 복잡한 프로세스를 계층적으로 분할하여 표현한다. 특히, 각 레벨 별로 나타나는 중복 정보를 최소화하여 각 추상화 수준에 맞는 정보만 표현되도록 한다.

**프로세스 개체간의 관계**: 프로세스 모델의 가장 중심이 되는 개체인 스텝은 다른 개체들과 다양한 관련성을 가지며 이를 모호하지 않게 자연스럽게 표현한다.

**순응적/반응적 제어 흐름**: 일반적인 스텝들의 실행 순서를 나타내주는 순응적(proactive) 제어 흐름뿐만 아니라 사건에 의해 스텝들의 상태나 실행 순서가 변경되는 모습을 나타내는 반응적(reactive) 제어 흐름을 표현한다.

**하나의 표현 프레임 워크**: 다양한 관점과 정보를 표현하기 위해 여러 모델링 도구를 사용하는 언어가 있다. Statemate 의 경우 행위도, 상태도, 모듈도 등을 작성해야 하며, UML 의 경우 열 가지 가까운 다이어그램을 작성해야 한다. 이러한 접근 방법은 비전문가의 사용을 매우 어렵게 한다.

## 4. RAiSE 의 표기법과 의미론

### 4.1 기본 개체

*스텝*은 누군가에게 할당된 작업을 말하며 RAiSE 모델의 가장 기본적인 구성 요소로 다른 개체들은 모두 스텝을 중심으로 연결된다. 스텝은 계층적 분할에 의해 또 다른 RAiSE 모델로 표현될 수 있다. *자원*은 한 스텝의 수행에 필요로 하는 개체를 말하며 각 스텝이 실행되기 전 획득되어야 하는 개체들을 표현하는데 사용된다. *산출물 저장소*는 산출물을 저장하는 물리적 장소를 나타내고, *조직*은 프로젝트에 관여하는 개인이나 팀을 말하며 스텝으로부터 특정 조직이나 개인에게 보내지는 메시지를 표현할 때 사용된다. *이벤트 바인더*는 이벤트를 처리할 때 사용한다. 모든 프로세스 모델은 *시작점* 심볼로부터 실행이 시작해서 *종료점* 심볼에서 종료된다. *메타점*은 메타 스텝이 실행되는 시점으로, 모델의 소유자가 이후 도달 가능한 스텝들에 대한 스케줄링 작업과 작업자 선정 작업을 수행할 수 있다. 시작점은 스텝의 실행 전에 반드시 만족해야 할 *선결 조건*을, 종료점은 스텝 종료 후 만족해야 할 *후결 조건*을 가질 수 있다.

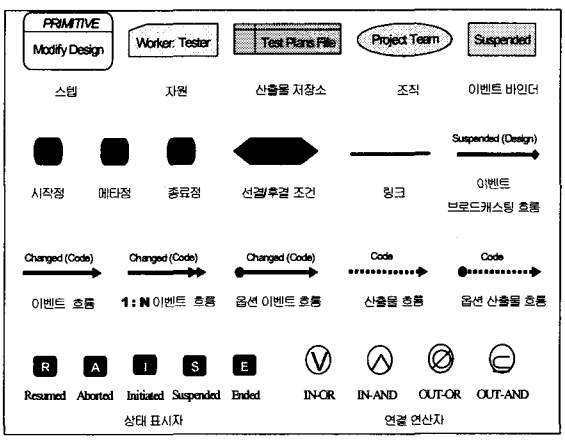


그림 1 RAiSE 표기법

### 4.2 개체간 연결

*링크*는 주로 자원과 스텝 사이를 연결하는데 사용된다. *산출물 흐름*은 두 개체 사이를 연결하는 점선 화살표 및 부착된 산출물 이름으로 구성되며 개체의 유형에 따른 동작 메커니즘 및 두 개체 사이에 전달되는 산출물을 표현하는데 사용된다. *이벤트 흐름*은 두 개체 사이를 연결하는 실선 화살표 및 화살표에 부착된 이벤트로 구성되며 발생한 이벤트를 메시지의

형태로 전달하고자 할 때 사용된다. 이 메시지를 이벤트 메시지라 하며 특히, 스텝과 스텝 사이의 연결은 모두 이벤트 흐름을 이용하여 표현한다. RAiSE 는 모델링의 일관성을 유지하고 작성된 모델의 이해도를 높이기 위해 자주 사용될 가능성이 높은 이벤트 메시지들에 대한 리스트를 제공한다.

### 4.3 스텝의 상태

스텝이 인스탄스화 되고 나면, 다섯 가지 상태 (Resumed, Aborted, Initiated, Suspended, Ended)[1] 중 하나에 놓여지게 된다. 스텝 상태는 스텝의 외부 요인 또는 내부 요인에 의해 전이되며 가능한 상태 전이는 모두 여덟 가지이다(그림 2). 각 상태에 해당하는 상태 표시자는 스텝에 부착되어 산출물 흐름이나 이벤트 흐름과 연결된다. 모든 스텝은 I 상태 표시자와 E 와 A 상태 표시자 중 적어도 하나의 상태 표시자를 갖는다.

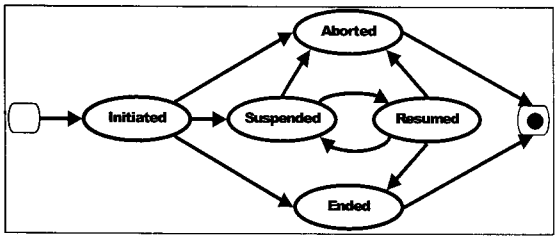


그림 2 스텝의 상태 전이

### 4.4 제어 구조

실제의 소프트웨어 프로세스는 반복, 선택, 병행, 결합 등 다양하고 복잡한 실행 순서를 내포하고 있으며, RAiSE 에서는 그러한 실행 순서를 다음과 같은 네 가지 *연결 연산자*를 이용하여 이해하기 쉬우면서도 효과적으로 표현할 수 있다.

IN-OR: 입력 흐름 중 하나라도 만족된다면 계속 진행한다.

IN-AND: 입력 흐름 모두가 만족된다면 계속 진행한다.

OUT-OR: 출력 흐름 중 일치되는 흐름으로 분기한다.

OUT-AND: 모든 출력 흐름으로 동시에 분기한다.

### 4.5 이벤트의 처리

이밴트란 스텝 실행에 영향을 주는 사건을 말하며, RAiSE 에서는 발생시키는 주체에 따라 모델에 표현된 이벤트 메시지에 의해 발생되는 메시지 이벤트와 시스템 클럭에 의해 발생되는 시간 이벤트로 구분한다. 이벤트를 표현하기 위해서는 이벤트의 발생과 처리 방법을 정의해야 한다.

메시지 이벤트는 *이벤트 브로드캐스팅 흐름*을 사용하여 발생시키며 *이벤트 바인더*를 통해 처리한다. 이벤트 브로드캐스팅 흐름에 나타나는 이벤트의 이름과 이에 대응되는 이벤트 바인더의 이름은 동일해야 하며, 이벤트와 더불어 전달되는 정보는 이벤트 바인더에 연결된 산출물 흐름의 이름에 바인딩된다. 즉, 이벤트 바인더에 연결된 산출물 흐름의 이름은 프로그래밍 언어의 형식 인자의 역할을 하기 때문에 동일한 이벤트에 대한 처리를 한 곳에서 수행할 수 있다.

프로세스 엔진에 의해 발생되는 시간 이벤트 역시 이벤트 바인더를 이용하여 처리하며 시간 이벤트는 절대 시작 시간, 절대 종료 시간, 상대 종료 시간을 나타낼 수 있다.

## 5. 적용 예

RAiSE 를 이용하여 ISPW-6 벤치마크 프로세스를 모델링한 결과의 일부가 그림 3 이다. ISPW-6 예제는 사용자 요구사항의 변경이 있을 경우 설계, 코딩, 단위 테스트를 수행하는 프로세스로 이 예제 프로세스를 RAiSE 언어를 이용하여 모델링한 결과는 총 여섯 개의 프로세스 모델로 나뉘는데, 그 중 소스 코드 변경 프로세스 모델을 그림 3 에 표현하였으며 소스 코드를 수정하고 컴파일하는 과정을 담고 있다. Modify Code 스텝이 시작되면, 먼저 Current Source Code 를 입력으로 받아 Modify Source 스텝이 실행되고 종료 후 Source Code 를 저장한 후 Compile 스텝이 실행된다. 컴파일이 실패하면 다시 Modify Source Code 스텝이 실행되며, 성공하면 Design Approved 이벤트가 발생할 때까지 기다렸다가 Manual Check 스텝이 실행된다. Manual Check 결과 코드와 설계 문서가 일치하면 성공적으로 종료되지만 일치하지 않으면 다시 Modify Source Code 스텝이 시작되어야 하며 이 때 메타점에서 스케줄링 등의 작업이 모델 소유자에 의해 이루어진다. 이 모델에 포함된 모든 스텝의 작업자는 Design Engineer 이다.

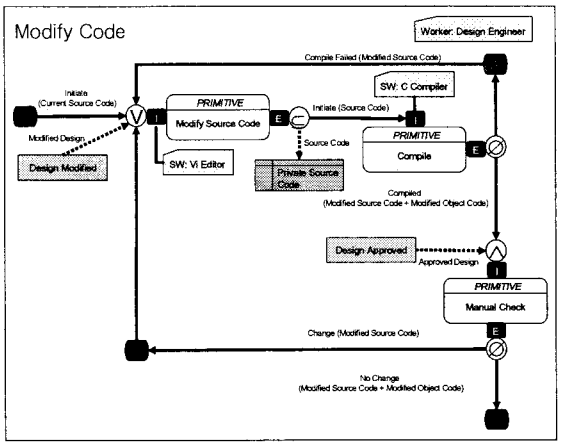


그림 3 코드 변경 프로세스 모델

다른 PML 들이 ISPW-6 프로세스를 모델링한 결과 [5,6,7]와 비교해 보면 RAiSE 는 높은 이해도와 다양하고 풍부한 의미론이 제공하는 프로세스 언어임을 알 수 있다. 특히, 프로세스 프로그래밍 개념에 기반한 그래픽 언어로 에이전트 조정에 매우 효율적이라

[1] 앞 글자들을 따면 본 PML의 이름인 RAiSE가 된다.

고 알려져 있는 Little-JIL 의 경우 70 개 이상의 스텝이 사용된 반면[7], RAiSE 의 경우 16 개의 스텝이 사용됨으로써 상대적으로 훨씬 간결하게 프로세스를 표현할 수 있다. 이는 Little-JIL 의 경우 제어 추상화와 프로세스 추상화라는 두 가지 추상화에 의해 프로세스가 계층적으로 분할되기 때문에 비단말 스텝이 많아지는 반면, RAiSE 의 경우는 프로세스 추상화에 의해 계층적 분할이 이루어지며 제어 추상화는 스텝들간의 제어 흐름으로 나타나기 때문이다.

## 6. 결론

본 논문에서는 사용의 용이성과 엄격하고 풍부한 의미론을 제공하는 PML 인 RAiSE 의 특징과 의미론을 기술하였으며 예제 프로세스에 적용한 결과를 통해 유용성을 입증하였다. 현재 RAiSE 모델을 실행시키기 위한 PSEE 인 PRAiSE 를 개발 중에 있으며 특히, PRAiSE 에서는 실행의 정형성과 효율성을 높이기 위해 전문가 시스템 개발 용 규칙 기반 언어인 Clips[8]를 이용하여 RAiSE 의 Rule 을 정의하고, 표현된 모델을 Fact 로 변환하여 실행시키는 방법을 사용한다[9,10].

향후 연구 과제로 지속적인 실제 프로세스 모델링을 통해 RAiSE 의 문법과 의미론을 계속 발전시켜 나가는 작업이 필요하며 프로세스 모델링 프로세스와 방법론을 제공하여 체계적인 프로세스 모델링이 이루어지도록 지원해야 할 것이다.

## 참고문헌

[1] Lerner, B. S., Osterweil, L. J., Sutton, Jr., S. M. and Wise, A., "Programming Process Coordination in Little-JIL," Proceeding of the 6th European Workshop on Software Process Technology(EWSPT'98), number 1487 in Lecture Notes in Computer Science, pp.127-131, Springer-Verlag, 1998.

[2] Sutton, Jr., S. M. and Osterweil, L. J., "The Design of a Next-Generation Process Language," Proceeding of the Joint 6th European Software Engineering Conference and the 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp.142-158, 1997.

[3] Kellner, M. I., Feiler, P. H., Finkelstein, A., Katayama, T., Osterweil, L. J., Penedo, M. H. and Rombach, H. D., "ISPW-6 Software Process Example," Proceedings of the 1st International Conference on the Software Process, pp. 176-186, 1991.

[4] Fuggetta, A. and Wolf, A., Software Process, John Wiley & Sons, 1996.

[5] Bandinelli, S. and Fuggetta, A., "Computational Reflection in Software Process Modeling: the SLANG Approach," Proceeding of the 15th International Conference on Software Engineering, pp. 144-154, 1993.

[6] Jaccheri, M. L. and Conradi, R., "Techniques for Process Model Evolution in EPOS," IEEE Transactions on Software Engineering, Vol.19, No.12, pp. 1145-1156, Dec. 1993.

[7] Lee, H., "Evaluation of Little-JIL 1.0 with ISPW-6 Software Process Example," Technical Report 99-33, University of Massachusetts, Computer Science Department, March 1999.

[8] Giarratano, J., "CLiPS Basic Programming Guide," http://www.ghg.net/clips/, 1998.

[9] 최상일, 이형원, "프로세스 중심 소프트웨어공학 환경의 구성요소간 인터페이스 설계에 관한 연구," 공업기술연구논문집, Vol.2, pp.321-325, Dec. 2002.

[10] 최상일, 이형원, 이승진, "프로세스 중심 소프트웨어공학 환경 PRAiSE 의 설계에 관한 연구," 제 5 회 한국 소프트웨어공학 학술대회 논문집, Vol.5, No.1, pp.219-228, Feb. 2003.