□ 기술해설 □

소프트웨어 프로토타이핑을 위한 CASE 도구

한국과학기술원 차 신* · 권용래**

목 차

1. 서 론

소프트웨어 공학분야의 꾸준한 연구결과로 요구되는 기능이 분명히 정의된 조건하에서 실행 가능한 코드를 제작하는 기술은 이제 상당한 수준에 이르렀다. 그러나 요구기능이 분명치 않을 경우에는 요구분석 및 설계 등의 개발 초기 단계에서의 잘못으로 완성된 소프트웨어가 사용자의 요구에 합당하지 않은 경우가 많아 사용자에게 인도된 후에도 많은 수정이 필요하거나 아예 인도되기 전에 폐기되는 사례도 있다[1]. 따라서 사용자의 요구를 명세서에 명료하게 반영하여 소프트웨어 개발팀이 명세서에만 의거하여 제작하였을 때에도 사용자의 요구에 부합되도록 하는 것이 소프트웨어 공학의 큰 과제중의 하나로 현재 많은 관심을 기울이고 있고 활발한 연구가 진행중이다.

서술적인 명세에 의한 소프트웨어 개발방법에서는 많은 노력을 기울여 만든 소프트웨어가 사용자의 요구와 맞지 않는 경우가 종종 발생한다. 이는 문자어가 갖는 애매 모호성으로 인하여 기술이 부정확하고 사용자와 개발팀간의 오해의 요소가 많기 때문이다. 구조적 방법론으로 대표되는 도식에 의한 명세 기법은 이러한 재래적인 명세서가 내포하는 문제점을 시정하기 위한 요구분석 기법이다. 특히 이러한 방법론을 지원하는 CASE 도구의 사용으로 이들 도식에 의한 소프트웨어 명세법은 요구분석 과정에서 어느정도 체계를 도입하여 요구명세의 정확성을 높이는 데에 큰 기여를 했다고 볼 수 있다.

이와 같이 도식에 의한 명세기법을 지원하는 CASE 도구를 사용함으로써 재래적인 명세서가 내포하는 문제를 줄일 수 있다. 하지만 요구분석의 관점에서 개발 방법론을 충실히 따라 시스템을 명세한다 할지라도 사용자는 자신이 어떠한 소프트웨어 시스템을 원하는 지를 분명히 모르는 수가 많고 설령 안다고 해도 그것을 정확히 표현하기 어려운 경우가 많다. 더욱이 소프트웨어는 비교적 추상적이어서 그것이 실제적 상황에서 작동하는 것을 관찰하기 전에는 명확한 견해를 갖기 어렵다. 원형화(prototyping)는 사용자의 기본적 필요를 포함하는 시제품을 단시일에 제작하여 실제적 상황에서 사용자에게 제시하고 작동시켜 보고 사용자의 반응을 수집하여 시제

* 정회원
** 종신회원

품에 반영시켜 가면서 반복적으로 확장, 개선하여 나가는 것이다[2,3].

이와 같이 요구분석 단계에서 실제적으로 유용한 도구가 되기 위해서는 요구분석 도구와 소프트웨어 원형화 도구가 효율적으로 통합된 CASE 도구가 되어야 한다. 본 고에서는 구조적 요구분석을 통하여 작성된 자료흐름도 명세서(또는 구조적 명세서)의 실행방식을 분류하고, 소프트웨어 요구분석 단계를 지원하는 CASE 도구중 구조적 요구분석 도구와 자료흐름도 명세서에 대한 원형화를 지원하는 CASE 도구에 대하여 살펴 보기로 한다.

2장에서는 소프트웨어 원형화의 정의와 분류 등을 설명하고, 3장에서는 자료흐름도 명세서의 실행방법과 자료흐름도의 수행모델에 관하여 기술한다. 6장에서는 message passing 방식을 자료흐름도 실행모델로 하여 소프트웨어 원형화를 지원하는 CASE 도구에 대하여 설명한다. 5장에서는 자료흐름도 명세를 직접 소스 코드로 변환하여 실행하는 방식에 대하여 기술하고, 6장에서는 순차 수행모델에 의하여 자료흐름도 명세서에 대한 원형화를 지원하는 CASE 도구에 대하여 기술한 후, 끝으로 7장에서는 결론 및 연구동향에 대하여 기술한다.

2. 소프트웨어 원형화

소프트웨어 원형화에 의한 방법의 주된 특징은 위험 요소의 감소, 원형의 진화를 통한 소프트웨어 개발 접근 방식을 들 수 있다[4,5]. 원형화 방식의 주된 이점은 문제의 성격과 그 문제에 대한 해결방법에 대하여 사용자의 요구사항들에 대한 불확실성을 감소시키는 데 있다. 소프트웨어 원형화를 통한 소프트웨어 프로세스 모델은 전통적인 방법과 달리 소프트웨어 요구분석에서 개발된 소프트웨어 원형을 통하여 시스템이 진화되어 가는 과정으로 소프트웨어 개발을 모델링한다[6].

원형화는 그 관점에 따라서 여러가지로 분류될 수 있다. 이를 다음과 같은 다섯가지의 관점에서 분류한다[7,8]

1) 문제해결 관점에서의 분류 : 설명형 원형화, 예비조사형 원형화, 실험적 원형화, 진화적 원형화
2) 원형화의 기능에 따른 분류 : 수직적 원형화, 수평적 원형화
3) 원형화 개발시기에 따른 분류 : 초기형, 중기형, 말기형
4) 개발 전략에 따른 분류 : 폐기형, 점진적인 개발
5) 개발형태에 따른 분류 : 제시형 소프트웨어 원형, 프로퍼. 소프트웨어 원형판, 파일롯트

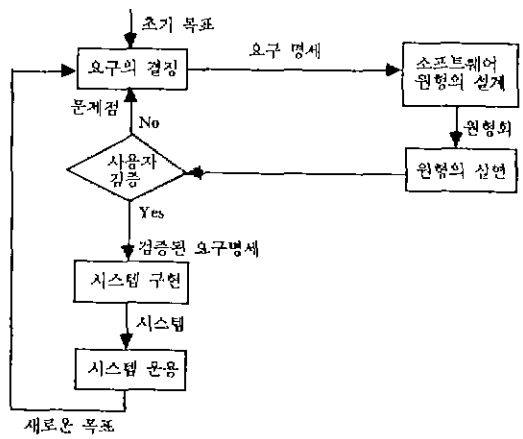


그림 1 원형화 과정

그림 1에 원형화를 이용한 소프트웨어 요구분석에 대한 과정이 나타나 있다.

사용자와 설계자는 계획된 시스템의 중요부분에 대한 요구명세를 정의한다. 설계자는 이를 토대로 원형기술 언어(prototype description language)를 이용하여 요구명세 수준에서 시스템의 원형을 구성한다. 구성된 원형은 단지 사용자의 특정한 요구를 반영하기 위한 전체 시스템에 대한 부분적인 표현이다. 소프트웨어 원형의 실연동안 사용자는 원형의 행동 특성을 평가하여 적절하게 동작하지 않으면 문제점을 식별하여 설계자와 요구명세를 재정의한다. 이러한 과정은 원형이 계획된 시스템의 중요한 특성에 대하여 적절히 동작할 때까지 계속된다. 설계자는 이와 같이 검증된 요구명세를 토대로 실제 소프트웨어를 설계한다.

원형화는 이해도를 증진시키며, 아직 개발되지

않은 시스템에 대하여 개발 의뢰자, 사용자, 개발자에게 동일한 시스템을 상정시킬 수 있게 하는 장점이 있다. 즉 소프트웨어 원형은 프로젝트에 참여하는 모든 사람들에게 동일한 개념모델의 기반을 제공한다. 또 소프트웨어 원형은 개발에 참여하는 모든 사람들의 의사 소통의 도구로 이용될 수 있고, 시스템이 개발되기 전에 요구의 변화가 프로그램에 어떤영향을 끼치는지를 검사하는 도구로 사용될 수도 있다.

3. 자료흐름도 수행모델

3.1 자료흐름도 명세서의 실행방식

구조적 분석기법은 하향식, 기능적 분해방법을 이용하여 시스템의 요구를 정의하는 모델링 기법이다. 개발하고자 하는 정보시스템을 프로세스의 집합으로 가정하고 각 프로세스와 다른 프로세스 또는 자료요소와의 관계를 자료흐름도로 표현하며 자료흐름도 상의 각 요소들에 관한 정의는 자료사전에 수집되어 수록된다. 자료흐름도와 자료사전으로 구성되는 시스템 모델이 완성되면 기초적 프로세스는 소단위명세서로 명세된다. 구조적 분석기법은 D. T Ross[9]에 의하여 처음 제안되었고, 현재는 Gane/Sarson과 DeMarco가 제안한 두가지 형태가 널리 알려져 있다.

자료흐름도는 사용자의 요구사항을 모델링하는 중요한 도구의 하나이다. 자료흐름도와 같은 그래픽한 형태의 도구는 표준화된 기호와 이를 해석하고 검증하는 정형화된 규칙을 제공하여 텍스트 형태의 명세가 주는 어려움을 극복하게 해준다. 그러나 프로그래밍 언어로 프로그램된 코드에서와 같이 컴파일 과정을 통해 내부적인 불일치성을 검증하고 이를 수행시켜 그 결과를 직접 볼 수 있는 것은 아니다.

자료흐름도 명세의 실행방법은 시스템의 행동양식을 보여주는 자세한 정도에 따라 세가지로 구분하여 설명될 수 있다.

3.1.1 스키마적인 수준(schematic level)

스키마적인 수준에서의 자료흐름도 명세의 수행은 시스템 모델의 전반적인 양식(pattern)이 옳은 지를 보여주는 데 목적이 있다. 즉, 시스템이 주어진 입력양식에 따라 질적인 측면에서 맞는 출력을 생산하는가를 보여준다. 하나의 프로세스 수행이라는 것은 단지 입력자료가 준비되어 있으면 수행되고, 수행결과는 출력 자료가 생성된다는 것이다. 이러한 형태의 수행은 petri net 수행방식으로 쉽게 설명할 수 있다. 토큰의 위치에 따라 수행상황을 알 수 있는데, 입력자료 흐름에 토큰이 있으면 수행할 준비가 되어 있는 것이며, 그 결과로 출력 자료흐름에 토큰이 생성된다. 이때, 입력자료 흐름에 있던 토큰은 제거되고, 출력자료 흐름이 다른 프로세스의 입력일 때 그 프로세스 역시 수행이 가능하다. 그림 2에 그 예를 보였다.

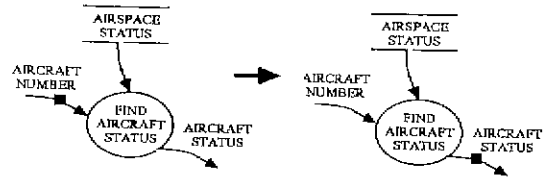


그림 2 간단한 프로세스의 수행 예

3.1.2 원형수준(prototype level)

원형수준에서의 자료흐름도 명세의 수행은 모델링되는 시스템을 질적인 측면에서 뿐만 아니라 양적인 측면에서 시뮬레이트한다. 즉, 수행 결과가 어떤 출력을 생산하느냐만 보여주는 것이 아니라 그 값까지 보여준다. 따라서 원형수준에서의 실행은 자동화의 관점에서 스키마적인 수준에서의 실행보다 앞서 있다고 말할 수 있다. 또한 여러가지 입력값을 시험해 봄으로써 출력값이 적절한지를 시험해 볼 수 있다. 이를 위해서 각각의 프로세스가 자료 변환에 관련되 명세를 가져야 하며, 명세를 기술하기 위한 상위 레벨의 언어가 필요하다. 응용분야에 따라 4세대 언어가 많이 도입되어 있으며, 원형을 지원하는 대부분의 CASE 도구는 프로그래밍 언어와 유사한 명세언어를 제공한다. 또한 실행을 위하여 초기값을 지정해야 하는데, 자료 저장소의 자료와 같은 저장되어 있어야 할 값들은 미리 준비되어야 한

다. 이러한 초기값은 대체로 데이타베이스 형태로 제공된다.

3.1.3 직접 구현적인 수준(direct implementation level)

직접 구현적인 수준에서의 수행은 자료흐름도 명세를 직접 해석하여 수행시키는 방식이다. 이 방식은 수행가능한 명세언어를 사용하는 방식과 수행모델을 따라 자료흐름도 구조를 직접 프로그래밍 언어로 번역하여 실행시키는 방법이 있다. 수행가능한 명세언어를 사용하는 경우는 시스템의 명세를 Z 등과 같은 수행가능한 명세언어를 직접 사용하는 것으로, 명세를 직접 수행시켜 보아 명세를 확정지어 가는 방식이다. 하지만 수행가능한 명세언어 자체가 응용영역이 국한되어 있기 때문에 일반적으로 적합한 방법은 아니다. 자료흐름도 명세를 프로그래밍 언어로 번역하는 경우는 LGDF 모델이나 message passing 방식을 수행모델로 하여, 자료흐름도 명세를 코드 변환을 위한 절차 및 각각의 절차들을 자동화하여 주는 도구들의 도움을 받아 C 코드 등과 같은 직접 수행가능한 코드로 바꾸어 준다. 이때, 프로세스들의 소단위 명세는 C 코드 등의 변환된 코드와 같은 언어로 기술되어야 한다.

3.2 자료흐름도의 수행모델

자료흐름도는 수행을 위한 형식적인 의미를 지니고 있지 않기 때문에 자료흐름도 명세서를 수행하기 위해서는 수행모델이 필요하다. 지금까지 개발된 자료흐름도의 수행모델로는 자료흐름도의 병렬성을 자동적으로 반영하기 위해 개발된 LGDF(Large Grain Data Flow) 모델, message passing 기법과 자료흐름도를 순차적으로 수행하는 방법을 들 수 있다.

Large Grain Data Flow 모델은 R. G. BabbII가 자료흐름도를 data driven 방식으로 구현하여 수행시키기 위하여 제안하였다[10]. 각각의 프로세스는 그 프로세스가 필요로 하는 모든 데이타가 사용 가능해지고 출력을 위한 공간이 생겼을 때에 계산이 시작된다. 따라서 자료흐름에 따라서 프로세스들은 비동기적으로 병렬수행이

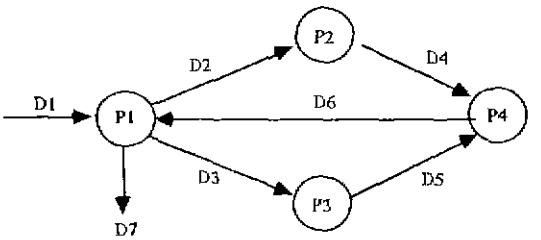


그림 3 순환이 존재하는 자료 흐름도의 예

가능하다. 그러나 이러한 수행모델의 LGDF 네트웍은 순환이 존재할 경우 본래의 자료흐름도와 의미상으로 일치하지는 않는다. 그림 3은 이러한 예를 보여주고 있다.

이러한 형태의 자료흐름도는 상황에 따라서 충분히 발생될 수 있는데 LGDF 네트웍 모델로는 프로세스들 사이에 자료흐름의 순환이 존재하므로 수행이 불가능하다. 근본적인 이유는 LGDF 네트웍의 수행모델이 자료흐름 계산모델을 기반으로 하기 때문이다.

병렬 프로그래밍 기법의 하나인 message passing 방식은 자료흐름도에서 병렬수행 가능성이 있는 프로세스들 사이에 자료의 이동을 자연스럽게 표현할 수 있다. 그림 3의 자료흐름도의 예를 들어보면 다음과 같다.

```
PROCESS(P1)
GET(INIT, D1)
.
.
PUT(P2, D2)
PUT(P3, D3)
GET(P4, D6)
.
.
END
```

프로세스 P1은 최초로 D1이라는 자료를 받아 처리하여 프로세스 P2, P3에게 자료 D2, D3를 보낸 뒤 프로세스 P4로부터 자료 D6가 보내지기를 기다린다. 그 이후에 프로세스 P4로부터 자료 D6가 전달되면 이를 받아 수행을 계속한다.

Message passing 방법을 이용한 자료흐름도의 수행은 자료흐름도가 가지는 병렬수행을 자연스럽게 표현할 수 있는 장점을 가진다. 그러나 mes-

sage passing 방법을 자료흐름도의 수행모델로 하였을 경우에는 프로세스의 과부하 상태로 인한 수행속도의 현저한 감소가 발생할 수 있다. Message passing 방식에서는 자료흐름도를 수행하기 위해서는 각각의 프로세스에 대하여 각각 하나의 프로세스를 할당하여야 하기 때문에 단지 몇개의 프로세스로 구성된 자료흐름도라 할지라도 수개의 프로세스가 동시에 활성화되어야 하므로 전체 시스템의 성능을 크게 저하시키게 된다. 따라서 실질적으로 사용하기에는 커다란 문제점을 가지고 있다.

자료흐름도가 지니는 병렬성은 일반적으로 프로세스들이 매우 복잡하게 연결되어 있는 상황에서 발생하는 것이어서, 앞에서 설명한 병렬 수행모델을 사용하더라도 병렬수행이 가능한 프로세스들을 병행성을 그대로 유지하면서 관리하기에는 많은 어려움이 따른다. 이를 해결하기 위하여 흔히 사용되는 방법이 순차 수행모델을 사용하는 것이다. 이 방법은 프로세스들의 수행순서를 미리 정하지 않고 사용자의 선택에 의한 유동적인 순서에 의하여 순차적으로 실행시키는 방법으로 수행순서의 결정은 수행 가능한 프로세스들의 범위를 동적으로 조정하는 방식으로 이루어진다. 계층 구조로 되어 있는 자료흐름도 상에서 어느 한 순간에 하나의 프로세스만이 수행이 가능하고, 이때 이 프로세스가 포함된 자료흐름도와 그 하위 레벨에 있는 자료흐름도 상의 프로세스들이 수행 가능한 프로세스들의 범위가 된다. 범위의 축소와 확대는 자료흐름도의 계층적 구조에 따라서만 이루어지며, 실제적인 실행의 의미를 지니는 최하위 프로세스에서 다른 최하위 프로세스나 이러한 방법은 요구분석 도구와 밀접하게 연관되고 프로세스의 선택을 위하여 사용자의 개입이 빈번하게 발생된다는 단점이 있다.

4. Message passing 모델에 의한 원형화 도구

여기서는 원형화 수준에서의 CASE 도구중 message passing 방식의 자료흐름도 수행모델을 사용하는 proto[11]에 대하여 살펴 본다. PROTO는 규모가 큰 소프트웨어 시스템에서 많은 소스 코드를 작성하기 전에, 시스템의 기능적 요구사항을 확인하고 검증하는 도구이다. 이를 위하여 PROTO는 다음과 같은 기능을 제공한다.

- 시스템의 기능을 나타내기 위한 자료흐름도 형태의 그래픽 언어
- 원형을 구성하기 쉽게 하는 재사용이 가능한 소프트웨어 모듈의 라이브러리
- 시스템 모델링과 원형 명세를 위한 에디터 및 에디터를 통해 만들어진 시스템의 기능적 요구사항을 실행시키는 인터프리터 등의 대화형 도구
- 원형을 구성하고, 분석, 검증하는 절차를 유도해 주는 방법론

PROTO에서는 자료흐름도와 재사용 가능한 모듈로써 시스템의 기능적 요구사항을 만든다. 이 요구사항을 인터프리터로 실행을 시키면 기능적 원형이 만들어지고, 시스템 설계자가 여러 개의 입력 시나리오를이 원형에 적용하여, 각각에 대한 출력을 검사함으로써 시스템의 요구사항을 평가한다.

PROTO가 제공하는 언어는 자료흐름도 형태의 그래픽 언어이며, 시스템의 각 요소들간의 계층적 구조와 객체 지향적 구조를 제공하고, 일반적인 자료 유형과 추상화 자료유형 개념을 제공한다. 또한 객체지향적 방법으로 데이타를 관리하여 계층구조로 되어 있는 자료흐름도와 자료 유형정의, 그리고 그 밖의 데이타들을 효율적으로 관리한다.

PROTO는 자료흐름도의 수행을 위해 message passing 모델을 사용한다. Message passing 모델에서는 각각의 프로세스들이 자신의 결과를 메시지의 형태로 출력하여 다른 프로세스들을 활성화시킨다. Message passing 모델에서는 시스템 설계자가, 연산의 실행 순서나 메모리 관리같은 구현에 관련된 부분에 관여할 필요가 없으므로, 상위 레벨의 언어보다 시스템의 요구사항을 나타내기가 편리하다.

PROTO에서는 각 요소들의 기능이 다음의 하나로 표현된다.

- PROTO의 자료흐름도
- 재사용 가능한 모듈

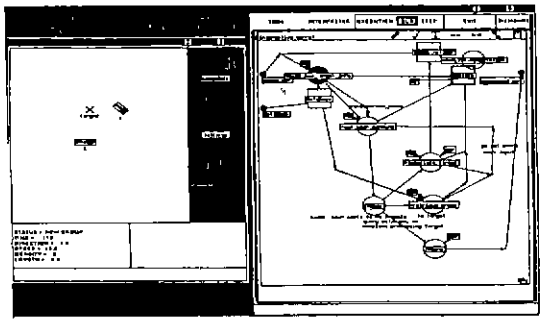


그림 4 PROTO의 사용자 인터페이스

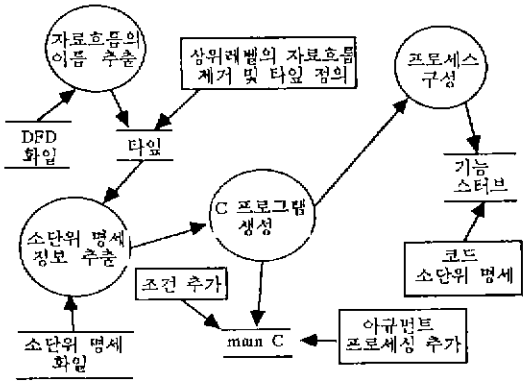


그림 5 SA Tool에서 C 소스 프로그램을 산출하기 위한 8가지 단계

—HHL 소스 코드 또는 script 언어

시스템의 기능적 원형은 자료흐름도를 인터프리트하여 실행되며, 한번의 실행때마다 주어진 입력에 대하여 시스템의 출력이 생성된다. 그러므로 여러개의 입력 시나리오를 시스템에 적용하여, 각각의 입력에 대한 결과를 가지고 시스템의 기능적 요구사항을 평가할 수 있다.

PROTO는 기능적 원형의 수행을 그래픽으로 사용자에게 나타내어 준다. 그림 4에서 오른편에 있는 자료흐름도를 보면 여러개의 프로세스들 중에서, 현재 수행되고 있는 프로세스는 검은색으로 강조되어 있으며, 왼편의 그림에서는 현재 수행되고 있는 프로세스의 상태와 바로 다음에 수행이 가능한 프로세스들을 보여주고 있다.

PROTO가 제시하는 방법론은 다음과 같이 다섯단계로 이루어진다.

1) 다음과 같은 시스템의 트랜잭션 모델을 정의 또는 변경한다.
   (가) 사용자의 요구나 명령어를 수행시키기 위해 필요한 모든 개체와 관계를 정의한다. 이들은 자료흐름도의 자료저장소에 해당한다.
   (나) 시스템의 기능을 시스템이 행해야 하는 트랜잭션의 관점에서 기술한다.
   (다) (가)와 (나)에서 구한 요소들을 재사용이 가능한 모듈을 이용해 표현한다.
2) 시스템의 사용자 인터페이스를 정의 또는 변경한다.
3) 트랜잭션 모델을 수행이 가능한 수준까지 자세히 기술한다.
4) 지금까지 정의된 트랜잭션 모델을 사용자 앞에서 실행하여 평가한다.
5) 변경되어야 될 시스템의 요구사항을 확인하여 1)부터 다시 시작한다.

5. LGDF 모델의 직접 구현 도구

이 장에서는 자료흐름도로부터 C 코드를 산출하는 Tektronix의 SA Tool[12]에 대하여 기술한다. Tektronix에서 개발한 자료흐름도로부터 C-소스 프로그램을 생성하고자 개발된 CASE 도구는 LGDF(Large-Grain Data Flow) 모델을 수행모델로 하고 있다. 자료흐름도상의 소단위 명세서는 독립적으로 수행가능한 모듈로 정의되고 메인 프로그램은 LGDF 모델에 따라서 이들 소단위명세서의 수행을 제어한다. 소단위명세서 입력 또는 출력되는 모든 자료 흐름에 대하여 하나의 플래그를 정의하면 다음과 같은 규칙에 따라서 각 소단위명세서의 수행이 제어된다.

1. 어떤 자료 흐름이 시스템 외부로부터 입력된다면 이러한 입력자료 흐름이 존재할 때 그 자료 흐름에 대한 플래그는 set된다.
2. 출력 자료 흐름이 산출될 때 그 자료 흐름에 대한 플래그는 set된다.
3. 입력 자료 흐름이 소모되면 그 자료 흐름은 clear된다.
4. 프로세스는 그 프로세스에 입력되는 모든 자료 흐름에 대한 플래그가 set될 때 수행되고 출력 자료 흐름에 대한 플래그를 set

한다.

구조적 분석 도구를 이용하여 산출된 자료흐름도로부터 C-소스 프로그램을 산출하기 위해서는 아래 그림과 같은 8단계를 수행한다. 그림 5에서 원으로 표시된 부분은 자동화가 이루어진 부분이고 사각형으로 표시된 부분은 수작업으로 수행되는 부분을 표시한다.

1) 자료 흐름의 이름 추출 : 모든 자료흐름도로부터 자료흐름의 이름을 추출하여 이를 저장한다.
2) 상위 레벨의 자료흐름 제거 및 타입 정의 : 추출된 자료흐름의 이름 중 소단위 명세서에 관련되지 않는 모든 자료흐름의 이름을 제거한 후, 나머지 자료흐름에 대하여 타입을 정의한다.
3) 소단위 명세정보 추출 : 구조적 분석을 통하여 소단위명세서에 기술된 내용을 최종 프로그램의 주석 형태로 삽입한다. 또한 자료흐름에 대한 레코드, 소단위명세서에 대한 이름 및 소단위명세서에 관련된 자료흐름 등의 레코드를 산출한다.
4) C 프로그램 생성 : 자료흐름에 대한 선언부, 자료흐름에 대한 플래그 선언부, 각 소단위명세서를 호출하는 메인 루프로 구성된 메인 프로그램과 각 소단위명세서에 대한 프로그램 스터브를 생성한다.
5) 프로세스 구성 : 각 소단위명세서 단위로 프로그램 스터브를 분할 한다.
6) 조건추가 : 어떤 소단위명세서는 모든 입력 자료 흐름이 가능하지 않아도 수행이 가능해야 한다. 이를 위해서 메인 프로그램에서 OR 조건을 삽입할 수 있도록 한다.
7) 코드 소단위 명세 : 단계 3에서 추출된 소단위명세서의 기능에 따라서 실제적인 C 언어로 프로그래밍한다.
8) 아규먼트 프로세싱 추가 : 메인 프로그램의 아규먼트의 추가.

6. 순차수행 방식을 이용한 원형화 도구

이 장에서는 동적제어에 의한 순차 수행모델에 의하여 자료흐름도 명세서에 대한 원형화를 지원하는 CASE 도구[13]에 대하여 기술한다. 그림 6에 전체시스템의 구조가 보인다. 요구명세 편집기는 구조적 분석방법을 지원하며 객체지향 방식의 사용자 인터페이스를 이용하여 사용자가 손쉽게 요구명세를 기술할 수 있도록 한다. 사용자는 이 편집기를 이용하여 자료흐름도, 자료사전, 소단위명세서로 구성되는 구조적 명세를 작성한다.

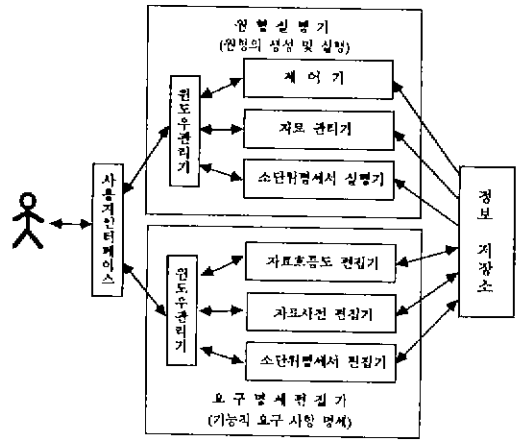


그림 6 시스템의 구성

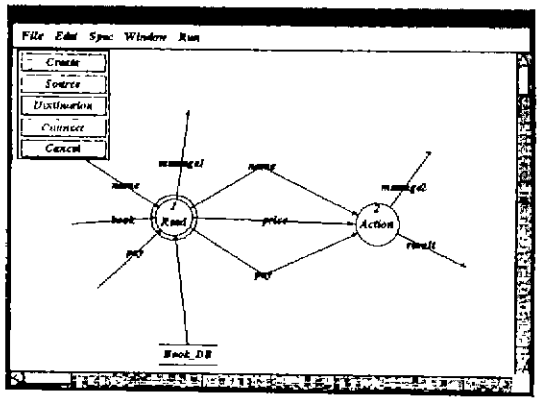


그림 7 자료흐름도 편집기

자료흐름도 편집기는 시스템의 전체구조를 구성하는 프로세스, 자료흐름도내에 유통되는 자료의 논리적 집합인 자료흐름, 자료를 보관하는 자료저장소, 시스템의 주변환경을 나타내는 외부객체 등의 구성요소들을 쉽게 표현할 수 있도록 하여 준다. 자료흐름도 편집도구는 하향식 분해화 기법을 지원하며 여러가지 표기법중 현재

표준모형으로 사용되고 있는 Tom DeMarco의 표기법을 따른다[14].

자료흐름도는 시스템 내에서 일어나는 자료의 변환과 프로세스간의 상호접속 관계를 표현할 수 있다. 그러나 자료항목들의 정확한 정의가 수반되지 않으면 단순히 시스템 내부에서 수행되는 업무에 대해서 어느정도의 정보를 전달해 주는 그림에 지나지 않기 때문에 모든 구성요소가 엄밀히 정의되어야 한다. 자료사전 편집기는 자료흐름 내용과 자료저장소 내용을 하향식 분할기법으로 정의할 수 있도록 하여 준다.

소단위명세서 편집기는 자료흐름도상에 나타나는 최하위 프로세스의 기능을 표현할 수 있도록 한다. 소단위명세서를 작성하는 도구로는 구조적 언어(structured language)나, 원형화를 지원하기 위해 설계된 원형기술 언어(PDL)가 사용된다.

시스템 분석을 효과적으로 수행하기 위해서는 위에서 설명한 세가지의 도구가 유기적으로 통합되어야 한다. 이 시스템에서는 자료흐름도 편집기, 자료사전과 소단위명세서를 기술하기 위한 편집기들이 정보저장소를 통하여 필요한 자료를 교환하며, 함께 사용할 수 있는 사용자 인터페이스를 제공한다.

그림 7은 자료흐름도 편집기를 이용해서 자료흐름도를 작성하는 모습을 보여준다. 자료흐름도 편집기는 pull-down menu와 pop-up menu로 구성된 인터페이스를 제공하는데 pull-down menu는 자료흐름도를 저장하거나 읽어들이는 등의 관리기능을 가지고 있고, pop-up menu는 자료흐름도를 그리는데 사용된다.

원형 실행기는 소프트웨어 요구분석 단계에서의 원형화를 지원하는 도구로서 요구분석가가 요구명세를 손쉽게 직접 수행하여 그 결과를 얻어 요구명세를 검증할 수 있도록 하여 준다. 자료흐름도의 수행모델로 동적인 제어를 통한 순차적 수행모델을 이용한다. 동적제어를 위한 정보를 사용자로부터 얻어야 하는데, 원형 실행기는 이를 위하여 여러가지 사용자 인터페이스를 제공한다. 사용자는 이러한 인터페이스들을 통하여 프로세스의 순서, 사용될 자료값 등을 지정하며, 원형 실행기는 이를 이용하여 요구명세

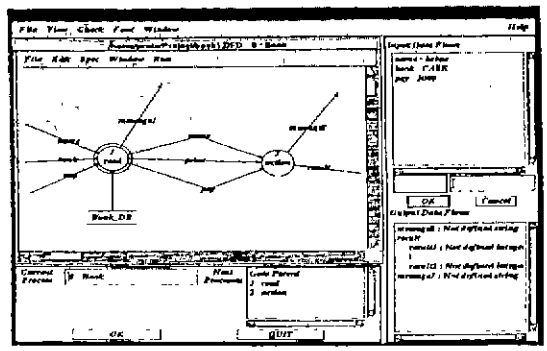


그림 8 동적 제어를 통한 원형화

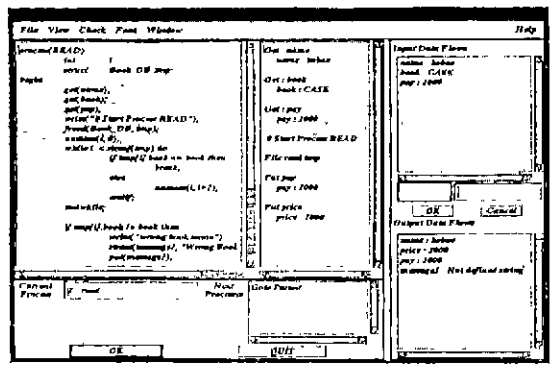


그림 9 소단위명세서의 실행

를 실행하고 그 결과를 보여준다. 사용자는 이 결과를 토대로 오류가 없으면 설계단계로 넘어간다. 만약 문제가 발생하였으면 요구명세를 변경하고 다시 원형화 과정을 반복한다.

그림 8과 그림 9는 원형화 도구의 사용 예를 보여준다. 원형화를 시작하게 되면 그림 8에서 보이는 것과 같이 현재 선택된 자료흐름도와 그 자료흐름도에 대한 입력 자료흐름, 출력 자료흐름들을 화면에 보여준다. 이때 사용자는 입력 자료흐름에 대해서는 원하는 값을 지정해줄 수 있다. 또한 현재의 자료흐름도에 포함되어 있는 프로세스들을 화면에 열거해주며 사용자는 열거된 프로세스중에 하나를 선택하는 작업을 반복함으로써 수행을 동적 제어하게 된다. 그림 9에는 사용자가 최하위 프로세스를 선택했을 때의 모습이 나타나 있다. 최하위 프로세스는 소단위명세서에 의해 그 기능이 기술되어 있으므로 원형화 도구내에 있는 소단위명세서 실행기에 의해서

그 명세가 실행되고 그 결과가 출력 자료흐름에 보여지게 된다. 또한 실행중에 발생되는 정보들을 사용자에게 보여준다.

7. 결론 및 연구 동향

본 고에서는 소프트웨어 요구분석 과정을 지원하는 도구로서 요구분석 도구와 원형화 도구가 통합된 CASE 도구에 대하여 살펴보았다. 이러한 도구를 이용하므로써 그래픽한 형태의 자료흐름도를 이용하여 사용자의 요구 사항을 명세하고, 소프트웨어 원형화를 통하여 요구사항을 명확하게 정의하는 것이 가능해지게 되며, 만들어진 원형은 최종 시스템에 대한 시제품으로 사용됨으로써 개발팀의 의사소통에도 도움을 줄 수 있을 것이다.

자료흐름도를 이용한 도식적 명세기법은 기존의 명세기법이 내포하고 있던 모호성을 해결하는데 많은 도움이 된다. 그러나 자료흐름도는 기본적으로 실행을 위한 형식적인 의미를 지니고 있지 않기 때문에 원형화를 지원하기에는 여러가지 문제점이 따르게 된다. 본 고에서는 자료흐름도 명세서에 대한 실행방법을 분류하여 설명하였고, 자료흐름도 수행모델에 대하여 기술하고 각각의 CASE 도구의 예를 살펴 보았다.

CASE 도구는 통합화되고 지능화되는 방향으로 발전하고 있다. 따라서 앞으로의 연구방향으로 실시간 시스템을 비롯한 대규모의 소프트웨어 개발을 지원하기 위해 다른 도구들과의 통합이용이하도록 도구의 통합방법에 대한 고려와 연구가 필요한다. 또한 각각의 자료흐름도 수행모델이 갖는 제약점들을 해결하여 자동화를 이루기 위한 연구가 필요하다.

참고문헌

[1] B. W. Boehm, Software Engineering Economics, Prentice-Hall, Inc. 1981.

[2] H. Gomaa, "The Impact of Rapid Prototyping," ACM SIGSOFT Software Eng. Notes, April 1983.

[3] B. H. Boar, Application Prototyping, Wiley-Interscience, 1984.

[4] Raymond T. Yeh, "An Alternative Paradigm for Software Evolution," Modern Software Engineering, Van Nostrand Reinhold. 1990, pp 7~22.

[5] Murat M. Tanik and Raymond T. Yeh, "Rapid Prototyping in Software Development," IEEE Computer, May 1989, pp. 9~10.

[6] Luqi, "Software Evolution Through Rapid Prototyping," IEEE Computer, May 1989, pp. 13~25.

[7] Tim Maude and Graham Willis, Rapid Prototyping, Pitman, 1991.

[8] H. Lichter, M. Schneider-Hufschmidt and H. Zullighoren, "Prototyping in Industrial Software Projects-Bridging the Gap between Theory and Practice," proc. 15th ICSE, 1993, pp. 221~230.

[9] D. T. Ross, "Structured Analysis(SA): A Language for Communcating Ideas," IEEE Trans, Software Eng., Jan. 1977.

[10] R. G. BabII, "Data-Driven Implementation of Data Flow Diagrams," proc. 6th ICSE, Sept. 1982. pp. 309~318.

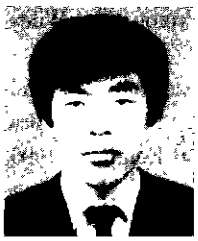
[11] Michael D. Konrad and Terry A. Welch, "Functional Prototyping with PROTO," Modern Software Engineering, Van Nostrand Reinhold, 1990, pp. 378~398.

[12] C. Olson, W. Webb and R. Weiland, "Code Generation from Data Flow Diagram," proc. 3rd Int'l Workshop on Software Specification and Design, Aug 1985, pp. 172~176.

[13] 통합형 CASE 도구의 설계 및 개발에 관한연구, 한국통신 장기기초 최종 연구보고서, 한국과학기술원 전산학과, 1993.

[14] Edward Yourdon, Modern Structured Analysis, Prentice-Hall, 1989.

차 신



1985 홍익대학교 공과대학 전자계산학과 졸업(학사)
1987 한국과학기술원 전산학과 졸업(석사)
1987 ~현재 금성중앙연구소 기초연구실 선임연구원
1992 ~현재 한국과학기술원 박사과정
관심 분야 : Software Metrics, Software Maintenance, Software Quality Assurance, CASE, Real-Time Systems

권 용 래

1969 서울대학교 문리과 대학 이학사
1971 서울대학교 대학원 이학석사
1971 ~1974 9월 육군사관학교 전임강사
1978 미국 피츠버그대학 이학박사
1978 ~1983년 7월 미국 Computer Sciences Corporation 연구원
1983 ~현재 한국과학기술원 전산학과 부교수
관심분야 : Software Specification, Development Tools, Programming Environment, Human-Computer Interface, Computer Supported Cooperative Works, CASE

● 동양언어 컴퓨터처리 국제학술대회(ICCPOL '94) ●

1994 International Conference on Computer Processing of Oriental Languages

동양 언어인 중국어, 일본어, 한국어를 포함한 동양 언어의 컴퓨터처리와 이해에 종사하고 있는 컴퓨터과학자, 정보공학자 및 시스템사용자들을 위한 국제적인 포럼을 목적으로 하는 학술대회가 금년 5월 대덕연구단지에서 개최됩니다. 특히 이제까지 중국어 컴퓨터처리를 중심으로 다루어 왔던 이 대회가 이번부터 모든 동양 언어로 범위를 확대하게 되었고 그 첫 학술대회가 한국에서 개최되는 데에 큰 의미가 있다고 하겠습니다. 국내에서 언어의 컴퓨터처리에 대한 관심은 아주 낮으며 이 분야를 체계적으로 연구하는 대학, 연구소, 산업체는 극소수에 지나지 않습니다. 이 대회가 이 분야에 대한 국내의 관심을 제고하고, 한국어 컴퓨터처리에 대한 연구 개발을 활성화하는 계기를 마련할 수 있기를 기대합니다.

- **기 간** : 1994년 5월 10일(화)~13일(금) (4일간)
- **장 소** : 과학문화센터(대덕호텔롯데) 대덕연구단지내
- **주 최** : 한국정보과학회 한국어정보처리연구회
  한국과학기술원 인공지능연구센터
  Chinese Language Computer Society
- **대회장** : 김길창(한국과학기술원 전산학과 교수)
  김영택(서울대학교 컴퓨터공학과 교수)
- **문의처** : 한국과학기술원 인공지능연구센터(Tel : 042-869-8713)