□신기술해설□

객체지향 CASE의 기술 동향

양 해 술† 이 용 근†† 이 하 용††

◆목 차◆

1. 서 언

객체지향 기술개발은 최근에 학계뿐만 아니라 산업계나 일반인들에게서도 언급되고 있는 신기술로써 많은 소프트웨어 개발자들이 아직 해결하지 못하고 있는 과제인 소프트웨어 위기를 극복할 수 있는 핵심적인 기술로 인식되고 있다. 현재 세계적으로 소프트웨어 개발자들은 운영체제나 응용 프로그램을 개발하는데 있어서 객체지향적인 기술을 접목시키려고 노력하고 있으며, 앞으로 다가올 21세기를 주도할 소프트웨어 개발의 기본 전략 기술로 자리매김하고 있다.

CASE는 소프트웨어 개발 과정을 지원하기 위한 자동화된 도구들을 의미하는 것으로 소프트웨어 개발의 자동화를 이룰 수 있는 도구이다. 현재의 CASE 도구들은 지원단계를 소프트웨어 생명주기 전과정으로 확대하여 각각의 부분적인 단계를 지원하고 있다. 이러한 CASE 도구들은 기존의 수작업으로 개발함으로써 발생하는 시간적/물적 노력을 절감하기 위해 다이아그램의 작성, 문서화 등을 자동화하여 실용적으로 이용할 수 있도록 지원하고 있다.

지금까지의 CASE는 주로 구조적 분석과 설계 방법론을 지원해 왔으나 요즘에는 객체지향 분석과 설계방법론에 대한 연구도 많이 이루어지고 있으며 객체지향의 장점이 인식되어 객체지향 분석과 설계방법론에 따른 소프트웨어 개발 전 과정을 지원할 수 있는 CASE 도구의 개발이 활발히 진행되고 있다. 구조적 기법이 실세계의 문제를 기능 중심으로 분석하여 점차 컴퓨터의 기능 중심 문제로 변환해 감으로써 소프트웨어 개발 기술을 공학의 관점에서 접근할 수 있는 중요한 기술이지만 요구하는 문제의 기능적 복잡성과 정보의 다양성이 더해감에 따라 한계가 드러나고 있다. 반면 객체지향 기술은 실세계의 문제를 객체 중심으로 해석하여 컴퓨터의 문제로 바꾸어 가는 과정으로 실세계의 현상을 가능한한 그대로 컴퓨터로 표현하고자 하므로 구조적 방법에서는 표현하기 어려웠던 복잡한 문제들의 표현 및 해결 뿐 아니라, 사후 유지보수에 있어서도 많은 장점을 가지고 있다. 따라서 본 고에서는 객체지향 CASE 도구의 개

† 종신회원 : 한국소프트웨어품질연구소 소장
†† 종신회원 : 강원대학교 전자계산학과 박사과정

발현황에 대해 살펴보고 대표적인 객체지향 CASE 도구인 COOAD를 설명하였다. 그리고 객체지향 CASE의 미래와 지적 CASE에 대한 요구 및 저장소에 대한 기술 동향을 기술하였다.

2. CASE 도구와 객체지향

2.1 CASE 도구의 실용화

CASE(Computer Aided Software Engineering) 도구는 소프트웨어공학이 제창된 이래 10년이상 계속해서 개발이 진행되어 왔지만, 지금에야 비로서 실용화 단계를 맞이하고 있다. 이것은 무엇보다도 현재의 소프트웨어 개발의 기간이 짧아지고 요구명세가 고도화됨에 따라 개발작업의 합리화가 큰 문제로 대두되었기 때문이다. 또한 다운사이징에 의해 CASE 환경을 구현할 수 있는 고기능 워크스테이션의 가격이 저렴해진 것도 보급의 관점에서는 빼놓을 수 없는 이유라고 할 수 있다. 즉, (그림 1)과 같은 환경에서 CASE에 대한 필요성이 높아지고 CASE가 실용적인 기능과 성능을 가지게 되었다.

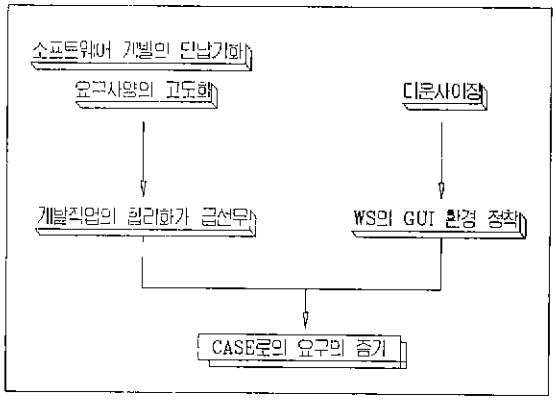

그림 1. CASE 도구 실용화의 배경

한편, CASE를 지원하는 분석·설계 방법론도 많이 개선되고 새로운 기법이 활발히 제안되고 있다. CASE를 실용적인 것으로 만들기 위한 방법론도 작감에 호소하는 복수의 관점을 가지고 동시에 각 관점에 적합한 표기법과 모델화의 지침이 명확히 되어 있어야 한다. 방법론의 보급도를 파악하는 척도로서 CASE 도구를 채용하는 정도라고 할 수 있을만큼 방법론 측면의 규칙에 대한 빈틈없는 대응이 반드시 필요하다고 본다.

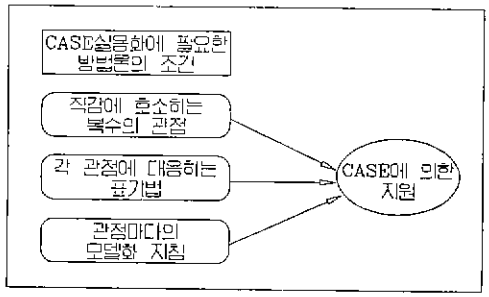

그림 2. CASE에 의한 방법론의 지원

2.2 CASE 도구와 객체지향의 이해 일치

객체지향 방법론을 지원하는 CASE 도구(객체지향 CASE)에 있어서도 여러가지 도구가 개발되어 상품화되고 있다. 지금까지의 방법론이 상위공정(분석·설계)을 지원하는 것에 불과한 반면, 객체지향 방법론은 상위로부터 하위공정으로 성과물을 잘 계승한다고 하는 큰 특징을 가지고 있다. 이러한 특징을 가지면 상위로부터 하위공정까지의 일관된 소프트웨어 개발환경을 실현하는 CASE 도구의 개발도 가능하게 되는 것이다. 한편, 새로 발생한 과제로서 Round Trip형의 개발 스타일을 얼마나 잘 지원하고 있는가를 예로 들 수 있다. 현재, 상용화되어 있는 각종 객체지향 CASE가 이와 같은 스타일을 지원할 수 있도록 고안되어 있다.

2.3 객체지향에 의한 객체지향 CASE의 개발

최근 Graphical User Interface(GUI)의 발달과 어플리케이션 소프트웨어에 사용되는 비율이 60%를 넘고 있다. GUI는 객체지향의 사고 방식에 기초하여 구축될 수 있었던 경위도 있고 필연적으로 객체지향 기술은 어플리케이션 개발에 있어서 중요한 사고 방

식이 될 수 있다. CASE 도구 자신도 이러한 어플리케이션의 하나로서 발달했으므로 객체지향 CASE를 구축할 때에도 객체지향의 사고방식이 매우 중요하다고 볼 수 있다.

3. 객체지향 CASE 도구의 현황

3.1 객체지향 CASE 도구의 분류

현재 객체지향 CASE 도구로 많은 제품이 개발되어 있으며 제품의 유형에 따라 다음과 같이 3가지로 분류할 수 있다.

3.1.1 객체지향 다이아그래밍 도구

객체지향 다이아그래밍 도구는 일반적인 객체지향 그래픽 표기법을 지원하며 하나 또는 그 이상의 객체지향 방법론을 지원하지만, 코드나 템플리트의 생성에 대해서는 거의 지원하지 않는다. 대표적인 도구로 EasyCASE, HOOD Toolset, Model 5w, Stood, TurboCASE 등이 있다.

3.1.2 객체지향 CASE 도구

객체지향 CASE 도구는 객체지향 다이아그래밍 도구의 기능인 그래픽 표기법을 지원하며, 한 두 가지의 특정한 객체지향 방법론을 지원한다. 그러나 단순한 다이아그래밍 도구와의 차이점은 응용 프로그램이 미약하지만 코드나 템플리트를 생성한다는 것이다. 현재 상품화되어 있는 대부분의 객체지향 CASE 도구들이 이 부류에 속한다. 대표적인 도구로는 Ada-Vantage, EiffelCase, MacAnalyst/Designer, ObjectCraft, Objecteering, ObjectModeler, ObjecTool, Object-Ory 등이 있다.

3.1.3 특수 목적용 객체지향 CASE 도구

실시간 응용프로그램들을 위한 CASE 도구들과 규칙기반(rule-based) 시스템 응용프로그램들을 위한 CASE 도구처럼 특별한 응용영역을 위한 CASE 도구들이 있다. 대표적인 도구로는 ObjectTime, ObjectGEODE 등이 있다.

3.2 객체지향 CASE 도구의 이용 현황

객체지향 기법은 어플리케이션 개발을 책임지는 기법으로 기대되고 있다. 이 기법을 소프트웨어 개발의 현장에 보급시키기 위해서는 이 기법을 지원하는 객체지향 CASE의 연구와 개발이 필수적이다. 현장에 공표되어 있는 각 기법은 그 방법론과 아울러 CASE가 제안되고 구체적인 도구도 나와 있다. 그러나 아직 현장에서는 이른바 「그림 그리는 도구」의 영역을 벗어나지 못한 것이 많으며 특히 상품화되어 있는 도구 중에는 원래 구조화 분석과 설계 기법을 지원하고 있던 도구였던 것을 기능을 확장시켜 객체지향을 지원할 수 있도록 한 것도 적지 않다. 이것은 구조화 분석에서의 데이타 흐름 다이어그램, 상태천이도, E-R도(객체관계도)이지만 객체지향에서도 그 표현형태로서 이용할 수 있기 때문이다.

3.3 객체지향 CASE의 실제

대표적인 개발 방법론과 그것에 대응하는 객체지향 CASE의 대응표를 〈표 1〉에 기술하였다. 우선 방법론을 지원하는 관점에서 살펴 보면, 예를 들어 OMT 기법을 지원하는 CASE로 GE사가 OMTool을 개발하고 있다. 이 도구에서 제공하고 있는 기능은 객체도의 편집기뿐이며 그 이외의 기능은 별도로 시판되는 도구를 이용해야 한다. 그 점은 다른 도구와 비교해서 부족한 점이긴 하지만 방법 자체의 완성도가 높기 때문에 다른 범용 CASE 도구 등에 채용하는 경향이 많다. Booch 방법을 지원하는 도구는 미국 Rational 사에 의해서 ROSE라는 이름으로 제품화되어 있다. 이 도구는 앞으로 기법을 전면적으로 지원할 것으로 되어 있지만, 현시점에서는 클래스도와 객체도의 편집에 이용할 수 있는 정도이다. 한편 주요한 방법론 대부분의 다이어그램을 지원하는 도구(예를들면 Object Maker)도 상품화되어 있다.

다음에 객체지향 CASE의 기능은 각 도구가 공통으로 지원하고 있는 주된 기능으로 여러가지 다이어

표 1. 대표적인 객체지향 방법론을 지원하는 CASE 도구

분석기법		Schlaer & Mellor 기법	Coad & Yourdon 기법	——	OMT 기법	Coad & Youidon 기법
설계기법		——		Booch 기법		OMT 기법
CASE 도구		Teamwork/OOA	OOATool OODTool	Rose	OMTool	ObjectCast
동작환경		UNIX WS	Macintosh, Windows, OS/2, UNIX WS	RS/6000, UNIX WS	UNIX WS	UNIX WS
기능	도형편집기	○	○	○	○	○
	프로토타이핑	△	×	×	×	△
	저장소	△	×	○	×	△
	단순한 역공학	○	×	×	×	×
	프로그램 생성	○	○	×	○	○
다른 도구와의 관계		——	OOD나 코드 생성 도구를 통합하고, OOWorkbench로 한다.	저장소로서 OODB의 [Varsant]를 이용	——	——
개발원 판매원		美 Cadre Technologies 社	美 Object International 社	美 Rational 社	美 GE 社	후지 Xerox 정보시스템

그램의 편집기능, 다른 다이어그램 간의 링크, 프로토타이핑, 저장소(Repository) 및 리포트 기능, 무결성의 체크 기능, 소스 코드 생성(C++, Ada), 역공학 등이라고 할 수 있다. 그리고 Round Trip 형의 개발에 대해서도 그렇게 지원하고 있는 도구들이 많다. 동작환경으로서 동작하는 하드웨어는 SUN 등의 워크스테이션이나 IBM-PC가 많고, 윈도우 시스템은 X-Window, Open Windows, MS-Windows 등이 있다. 다음 절에서는 한가지 사례를 이용하여 객체지향 CASE의 기능을 좀 더 상세히 살펴보기로 한다.

4. 객체지향 CASE COOAD의 실제

4.1 분석에서 프로그램 생성까지 일괄 지원

하나의 사례로서, 객체지향 CASE의 프로토타입인 COOAD(CASE tool for Object-Oriented Analysis and Design)를 소개한다. COOAD의 특징은 객체지향 분석과 설계의 각 단계를 일괄적으로 지원함과 동시에 이들 결과로부터 C++의 골격까지 출력할 수 있는 것이다. COOAD에서는 (그림 3)과 같이 4 단계로 명세화 프로세스를 실행한다.

단계 1부터 3은 분석·설계의 공정이며 여기에서 단계 번호는 기본적인 작업순서를 나타내고 있다. 그러나 이들 단계 간의 이동은 자유로우며 이른바 Round Trip에 의한 분석·설계가 가능하게 되어 있다. 이들 공정에서 얻어진 분석·설계 정보는 저장소에 저장되고 이것으로부터 최종 성과물인 C++의 골격 코드가 얻어진다.

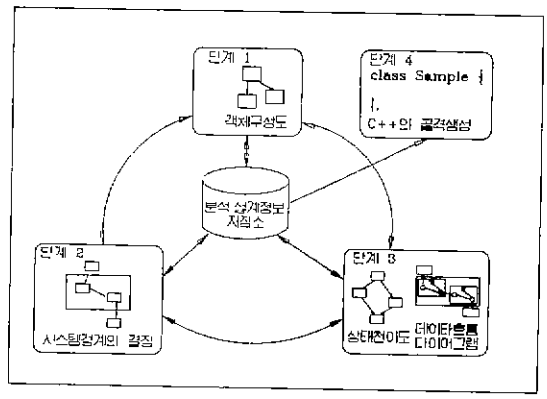

그림 3. COOAD의 구성

4.2 각 단계간의 Round Trip을 지원

단계 1에서는 객체를 추출하고 그것들 간의 관계를 객체 구성도를 이용하여 정의한다. 각 클래스에서

는 클래스명, 속성명, 조작명 및 그들의 형을 정의한다. 클래스간에는 "is-a" 관계, "part-of" 관계, "association", 관계의 3종류를 정의한다.

단계 2에서는 시스템 경계를 결정한다. 여기에서는 시스템 경계의 내외에 객체를 배치하고 객체간 메시지의 흐름을 자세히 기술한다.

단계 3에서는 단계 2에서 시스템 경계내에 배치된 개개의 객체를 상태천이도와 데이타흐름 다이어그램을 이용하여 기술한다. 상태천이도는 각 객체마다 기술한다. 이 그림에서는 객체에 대한 각 조작에 대해서 그 객체가 해야 할 행동(action)이나 상태의 변화를 표현한다. 데이타 흐름 다이어그램은 단계 2를 상세화한 것이고 각 조작이나 속성간의 데이타 흐름을 기술한다. 이 그림에서는 조작은 버블(bubble)로서 속성은 데이타 스토어로서 표현한다. 어떤 경우에는 버블은 재구성되지만 COOAD에서는 미리 결정된 규칙에 기초한 것을 지원한다.

단계 1의 객체 구성도, 단계 2의 시스템 경계도의 정보 및 단계 3의 상태천이도나 데이타 흐름 다이어그램은 여러가지로 밀접하게 관련되어 있다. COOAD는 각 그림을 그릴 때 다른 그림에 기술되어 있는 정보를 이용할 수 있도록 지원하고 이 그림들간의 무결성을 지킨다. 예를 들면, 데이타 흐름 다이어그램을 그릴 때에는 객체 구성도에서 추출한 속성이나 조작을 이용할 수 있다. 또한 역으로 데이타 흐름 다이어

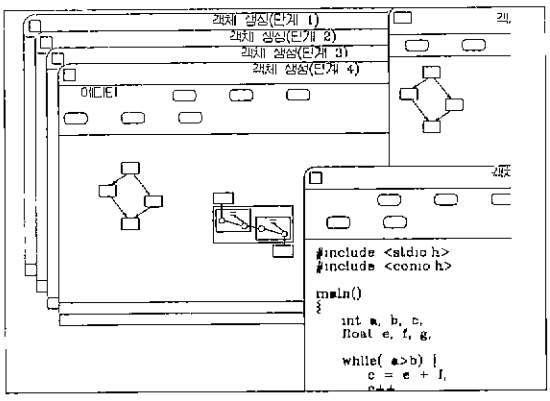

그림 4. COOAD의 화면 메시지

그램에 있어서 추가나 수정은 객체구성도에 자동적으로 반영된다.

단계 4에서는 지금까지의 단계에서 생성된 클래스의 명세에 기초하여 바라는 프로그램의 골격을 출력한다. 설계자는 이 골격에 덧붙여 필요한 기능을 추가함으로써 최종적인 프로그램을 얻는다.

5. 객체지향 CASE의 미래

5.1 다이아그래밍 도구의 탈피

현장의 CASE 도구들이 지원하고 있는 것은 ① 도형 편집기, ② 프로토타이핑, ③ 프로그램 생성, ④ 저장소, ⑤ 단순한 역공학 등의 기능이다. 많은 표기법의 지원이 중심이고 일부에 해석계가 존재하지만 다이아그래밍 도구의 역할을 완전히 벗어났다고는 말하기 어렵다. 따라서 이들 도구를 사용하기 위해서는 객체지향분석과 설계 방법론을 숙지하고 있어야 한다. 예를들면, 그들 도구에서는 클래스와 다른 클래스와의 정적 및 동적 관계를 표기하는 객체관계도를 쓸 수 있는 것처럼 되어 있지만 무엇을 클래스라고 하는가라는 관점에서의 지원은 아니다. 사용자의 입장에서 컴퓨터로 지원해 주었으면 하는 것은 어떠한 상황에서 과거의 성공사례 또는 실패사례나 클래스 추출의 기준을 보이는 것이다. 즉, 소프트웨어 개발 프로세스의 여러가지 상황에서 이와 같은 지적 부분의 지원이 요구되고 있다.

5.2 지적인 소프트웨어 개발환경의 제공

이제부터의 객체지향 CASE는 지적인 소프트웨어 개발환경이 되어야 한다. 지적인 소프트웨어 개발환경(이하, 지적 CASE)이라는 것은 인간과 컴퓨터의 협조작업을 효율성있게 지원하는 환경이라고 볼 수 있다. 이와 같은 환경에서는 모든 작업을 컴퓨터가 하기 때문에 본질적으로 지적인 작업만을 인간이 하고 컴퓨터는 그 지적인 작업을 보조하고 유도하는 등의 여러가지 잡무만을 담당한다. 다음 절에서는 지적 CASE에 대해 살피보기로 한다.

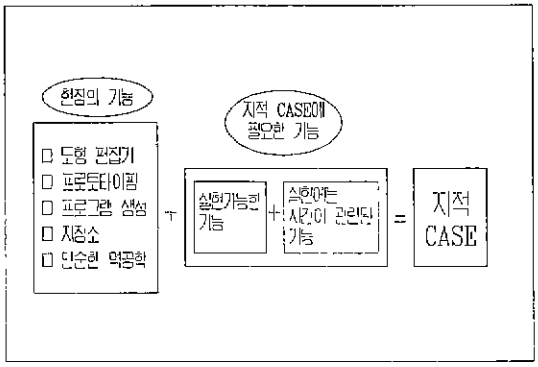

그림 5. 지적 CASE에 포함된 객체지향 CASE 기능

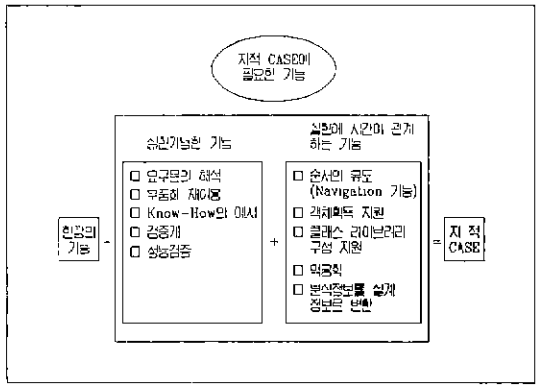

그림 7. 지적 CASE에 필요한 기능

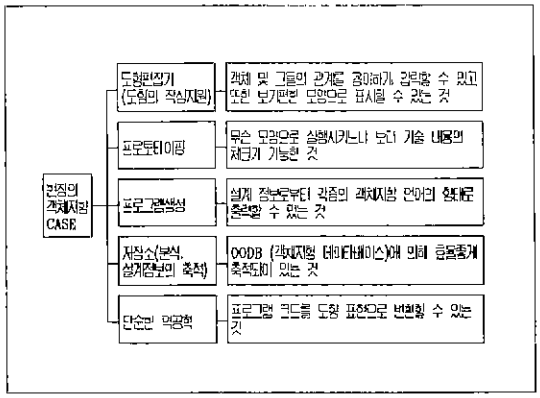

그림 6. 현장의 객체지향 CASE의 기능

5.3 지적 CASE에 대한 요구

(1) 단순작업을 경감시키는 기능

지적 CASE에는 설계자의 수행 작업을 정확하고 효율적으로 지원할 수 있도록 단순작업을 경감시키는 기능이 필요하다고 생각된다. 앞 절에서 기술한 바와 같이 객체지향 CASE가 각종 표기법에 의한 명세 기술을 지원하기 위한 도형 편집기 기능이라고 하면 지적 CASE에 실현해야 할 기능은 명세기술 이외의 분석·설계 작업을 지원하는 기능이라 할 수 있다.

(2) 실현가능한 기능

가장 실현 가능한 기능부터 살펴보면, 우선 요구문의 해석이지만 명사구를 정리하고 동명사구를 중심으로 동작주체, 동작대상을 선출하는 것은 충분히 할 수 있는 것이다. 프로토타이핑은 분석에 의해 명세를 어느 정도 확정하는 단계로 상세설계 단계에 이르기까지를 지원하는 것이다. 검증계도 프로토타이핑과 거의 같은 공정에서 이용하지만 양쪽의 내용은 다르다. 검증계는 영어로 Verification(검증), 프로토타이핑은 Validation(확인)이라고 부르는 것처럼 전자는 정확한지의 확인을 컴퓨터가 처리하는 것이고, 후자는 확인하는 것이 인간의 행위이다. 이 양자에 의해서 작업 프로덕트를 올바르게 확인할 수 있다. 상세설계 정보가 올바르다고 검증·확인되면 그로부터 프로그램을 자동생성한다. 현장의 프로그램 생성기능은 대상이 되는 프로그램의 골격을 생성하는 것에 불과하며 사용자는 생성된 골격에 수동으로 프로그래밍하지 않으면 안된다. 이것은 보수의 관점에서 효율적이 못되고, 가능하다면 완전한 프로그램을 생성하고 싶은 것이다. 위에서 보인 검증이라는 것은 기능의 검증이다. 실시간 시스템 등에서는 프로그램의 성능이 엄밀히 제약되고 있다.

따라서, 대상이 되는 프로그램이 성능요구를 만족하고 있는가를 가능한한 초기 단계(상세설계 단계가 타당)에서 검증하고 필요에 따라 기능 설계에 반영할 필요가 있다. 이 때에는 이 프로그램이 수행하는 OS나 하드웨어의 정보에 입각하여 성능을 검증할 필요가 있다.

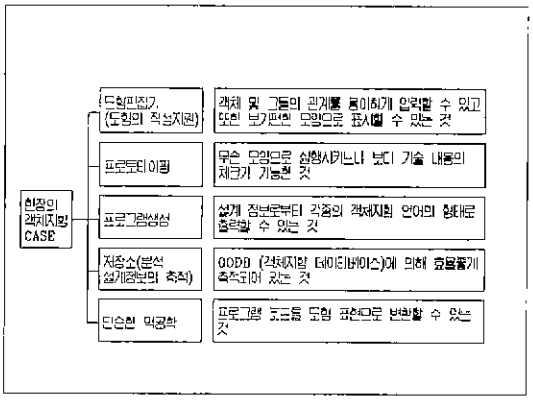

그림 8. 지적 CASE로서 실현 가능한 기능

(3) 재사용 지원과 Know-How의 예시

재사용 지원과 Know-How의 예시에 대해서는 아직 축적한 내용을 수집하고 있는 단계로서 재사용에서는 아직 어떤 기능이 필요하게 될지 연구가 불충분하고 이점이 앞으로의 중점과제이다. Know-How의 예시는 교과서에 보이고 있는 Know-How를 표시하는 정도의 간단한 기능이라면 바로 실현할 수가 있다. 그러나 그것으로는 그다지 실용적이지 못하므로 Know-How를 축적하여야 할 것이다.

6. 지적 CASE에 대한 요구 (2)

6.1 객체의 획득지원, 클래스 라이브러리 설계지원 기능

객체의 획득지원 또는 클래스 라이브러리의 설계지원 기능은 실현에 많은 시간이 걸린다고 하더라도 가장 중요한 것으로 생각된다. 객체지향 소프트웨어 개발 기법은 객체를 핵심으로 하여 작업을 진행한다. 그렇기 때문에 객체의 추출 여부에 따라 명세의 좋고 나쁨이 결정되어 버린다. 객체 추출 작업에 관해서는 방법론 중에서는 너무나 많은 것을 기술하고 있지만 단지 명사를 추출하여 그 중에서 택한 것을 지적하는 것으로 그치고 있다. 객체의 추출에서는 여러가지 판단기준으로부터 종합적으로 판단할 필요가 있고 객체지향 CASE에서는 이 판단기준을 안내하는 기능이나 명세서로부터 뺀 명사 그리고 명사구를 자동 분류하는 기능 등이 요구된다. 아울러 이들 추출된 클래스가 적당한 것인지를 평가하는 도구도 필요하게 된다.

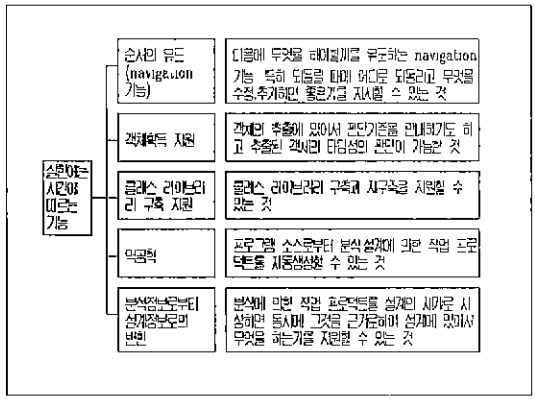

그림 9. 지적 CASE로서 실현에 시간이 필요한 기능

객체 획득은 객체지향 설계의 기본이 된다. 즉, 소프트웨어의 기본설계 담당자는 최종적으로는 시스템에서 사용하는 클래스를 정의하고 그것을 클래스 라이브러리로 통합하고 각 객체의 상세설계 담당자에게 제공한다. 일단 구축된 클래스 라이브러리는 Round Trip형의 소프트웨어 개발에 의해 빈번히 재구축이 강요된다. 따라서 클래스 라이브러리의 구축과 그 관

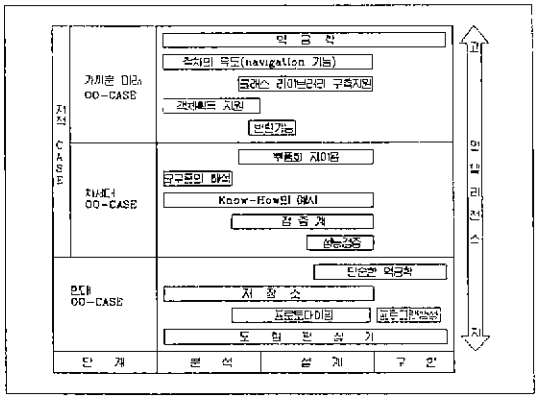

그림 10. 지적 CASE에 의한 작업의 흐름

리는 실제 프로그래밍 작업에서는 가장 중요한 작업이 된다.

6.2 Navigation 기능

순서(객체지향 분석·설계 프로세스)의 유도, 즉 다음에 무엇을 해야 할 것인가를 유도하는 Navigation 기능은 지적 CASE에서의 최종적인 과제라고 할 수 있다. 현장의 기법에서는 자연스런 순서가 확립되어 있다고 말할 수는 없지만 상세한 명세화를 가능하게 하는 것이다. 특히 되돌릴 때에 어디로 되돌려 무엇을 수정하고 추가하면 좋은가를 지시할 수 있으면 Round Trip형의 개발은 용이해지며 상세한 명세화 설계 프로세스를 정형화하고 그 프로세스에 따라 작업을 해 나가려면 아래의 항목을 명확하게 할 필요가 있다.

(1) 객체의 후보를 찾아내는 방법과 그 중에서 타당한 객체를 선택하는 방법에 대한 상세한 기준과 순서
(2) 객체 자신을 상세히 기술하고 다른 객체 사이의 정적 및 동적인 관계의 기술을 진행하는 순서
(3) 객체간의 관계를 나타내는 구조를 어떻게 구축하고 재구축해야 할지의 기준 및 순서
(4) 객체지향 분석에서 얻어진 생성물을 객체지향 설계의 세계로 사상(mapping)하는 순서

7. 저장소(repository)

7.1 저장소의 기능

7.1.1 기본 기능

저장소에는 정보를 저장하거나 검색하기 위해 다음과 같은 기능들이 필요하다.

① Common Items
프로젝트에서 공통적으로 사용할 자료를 등록하여 이용하는 기능

② Domains & Base Types
상품화된 관계형 데이타베이스 관리시스템에서 제공하는 정의역과 기본 자료형을 생성하는 기능

③ 참조 기능
WRM 저장소에 저장된 정보들을 이용할 수 있도록 지원하는 기능

④ 자료사전 관리 기능
자료사전의 생성, 열기, 닫기, 저장하기 등의 기능

7.1.2 변경관리 기능

방법론 지원도구와 WRM 저장소 간의 불일치를 막기 위해 동기화시키는 기능으로 다음과 같은 기능들이 필요하다.

- 도구별 저장소 수정 기능
  WRM 저장소의 정보를 기준으로 도구별 저장소의 정보를 수정하는 기능
- WRM 저장소 수정 기능
  도구별 저장소의 정보를 기준으로 WRM 저장소의 정보를 수정하는 기능

7.1.3 변경효과 추적 기능

변경관리를 하기에 앞서 변경의 효과가 무엇인지 사전에 알아볼 수 있도록 지원하는 기능이다.

7.1.4 외부접속 기능

다른 CASE 도구와 정보를 교환할 수 있도록 지원하는 기능이다. SILVERRUN에서는 ADW/IEW와 자료사전을 교환할 수 있도록 지원한다. 또한 Informix, Sybase, Oracle, Ingres 등 10여 개의 관계형 데이타베이스 관리시스템과 연결하는 기능이 있으며 IBM A/S 400의 4GL Synon과도 연결된다.

7.2 지적 객체지향 CASE 저장소

저장소는 분석 및 설계작업의 성과물을 효과적으로 저장하는 구조로서 객체지향 CASE의 경우에는 객체 모델, 동적 모델, 프로세스 모델을 서로 관계없이 저장하게 된다. 이와 같이 현장의 저장소의 다수는 도형으로 표현된 명세를 대상으로 하지만 지적 객체지향 CASE의 중요한 기능인 Know-How를 예시하는 기능을 실현하기 위해서는 단순한 성과물에 그

치지 않고 이유, 성공사례, 실패사례 등의 정보도 저장해야 한다. 이를 위해서 우선 사례를 범용화하고 Know-How를 표준화하는 방법을 찾는 작업이 필요하게 된다. 그러한 정보는 점점 증가하기 때문에 검색효율이나 유사검색 기능이 충실해야 하고, AI 기법의 적용이 필요하게 된다. 예를 들면, 사례 베이스 추론은 그를 위한 유력한 구조이다. 유사검색기능은 효율적인 부품화 재이용 기능의 실현에 있어서도 필요하다. 아울러 기능 향상을 도모하고, 객체획득 기능이나 클래스 라이브러리 구축지원 기능을 실현하기 위해서는 Know-How를 이용한 문제해결 메카니즘의 구현이 필요하다.

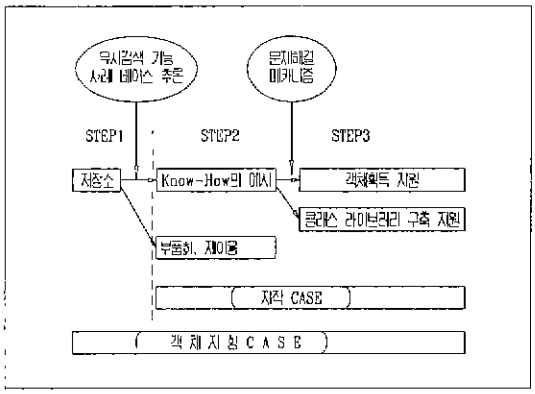

그림 11. 지적 객체 CASE를 지원하는 저장소

7.3 표준화가 진행되는 저장소

저장소가 제안된 당초보다 그 중요성에 대한 인식이 높아지긴 했지만 실현 레벨에서는 기능면, 성능면에서 충족할 수 있는 레벨로 완성되지 못했기 때문에, 아직 결정적인 것은 나타나고 있지 않다. 특히 지금까지의 저장소에서는 자사의 CASE 용에 한정된 것으로 실현되어 있기 때문에 타사의 유용한 도구가 사용될 수 없다는 문제점이 생겼다. 최근에 이것을 해결하기 위해서 표준화 작업이 진행되고 있고 나중에는 표준에 따르는 형태로 정보 저장소를 고려할 필요가 있다.

표준화의 동향에서 최근 가장 주목되고 있는 것은 ECMA(European Computer Manufactures Association)에서 제안하고 있는 PCTE(Portable Common Tool Environment)일 것이다. 이미 많은 유력 업체가 이것을 채용할 움직임을 나타내고 있고 앞으로의 주류가 될것으로 주목되고 있다. PCTE는 토스터 모델을 채용하고 있고 각 도구는 도구 슬롯이라 불리는 영역에 집어 넣음으로써 저장소를 경유하여 인터페이스를 취하는 것이 가능하다. 즉, PCTE를 사용하게 되면 LAN상에 위치한 워크스테이션들은 저장소가 분산되어 있는 것처럼 다른 워크스테이션들과 정보교환이 가능하다.

PCTE는 원래 구조화 분석·설계 기법 등 기존의 기법을 기반으로하여 고려되고 있기 때문에 객체지향 CASE를 만들려면 더욱더 새로운 기능이나 인터페이스가 필요하다고 생각된다. 앞으로는 첫째로 이들 기능의 명세를 결정하고 그후 객체지향 데이타베이스(OODB)에서의 실현법을 검토할 필요가 있다.

객체지향 CASE에서 저장소는 OODB로 실현하는 것이 적당하다고 생각된다. 실제, 객체를 효율좋게 관리하는데는 종래의 관계 데이타베이스에서는 만족한 성능은 얻기 어렵고 특히 처리속도나 데이타의 형이 프로그래밍 언어와 데이타베이스간에 일치하지 않는 점 등이 문제가 된다.

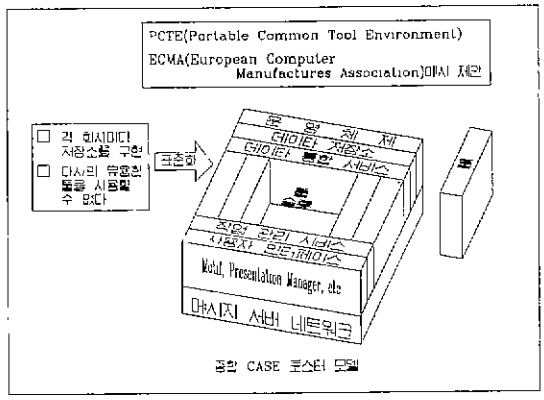

그림 12. 저장소 표준모델로서 주목되는 토스터 모델

8. 결 언

지금까지 객체지향 CASE에 관한 전반적인 기술 동향을 살펴보았다. 실세계의 현상을 그대로 컴퓨터에 적용하여 문제를 해결하고자 하는 객체지향의 사고방식을 자연스럽게 지원할 수 있는 객체지향 CASE의 개발은 소프트웨어의 품질에 대한 인식이 높아지고 유지보수의 문제가 중요한 관건이 되고 있는 현시점에서 대단히 시급한 문제라고 할 수 있을 것이다.

이러한 관점에서 객체지향 CASE 도구는 다음과 같은 기능들을 기본적으로 갖추고 있어야 한다고 볼 수 있다. 첫째로 객체지향 CASE 도구는 기본적으로 객체지향 개념을 위한 적절한 다이어그래밍을 표현하는 기능을 가지고 있어야 한다. 아울러 사용자의 필요에 의해서 그래픽 표현이 조정될 수 있는 유연성이 있어야 한다. 둘째로 객체지향 CASE는 모형의 일부분을 보이지 않도록 하거나 다시 보이도록 하는 기능을 가지고 있어야 한다. 객체지향 모형은 매우 복잡하기 때문에 모형에서의 일부분을 사용자가 원할 때 감추거나 다시 보여주는 기능이 매우 유용하다. 셋째로 객체지향 CASE는 검색기능을 갖추고 있어야 한다. GUI 환경에서 여러 개의 윈도우를 열어서 정보저장소에 저정되어 있는 요소들의 속성이나 요소들 간의 관계, 다이어그램 간의 항해 등이 검색 편집기 또는 검색 언어를 사용하여 다양하게 검색할 수 있도록 지원되어야 하기 때문이다. 넷째로 객체지향 CASE는 일치성 및 오류 검사 기능을 가지고 있어야 한다. 규모가 크고 복잡한 시스템의 모형에 대한 오류 검사 기능은 CASE 기술을 활용하게 하는 이유 중의 하나이기 때문이다. 다섯째로 객체지향 CASE는 그룹웨어를 지원해야 한다. 여섯째로 객체지향 CASE는 코드생성 기능을 지원해야 한다. 일곱째로 객체지향 CASE는 다른 도구나 정보저장소 등과 통합하여 사용할 수 있는 환경을 지원해야 한다.

이상에서 객체지향 CASE 도구가 기본적으로 지원해야 할 기능들을 살펴보았다. 그러나 현재 개발되어 있는 CASE 도구들 중에서 이러한 요구사항을 만족하고 있는 도구는 극소수에 불과하며 아직 부족한 점이 많이 있다. 부족한 점들로는 첫째로 현재의 도구들은 방법론이 지원하는 기법보다는 그래픽 표현 자체에 중점을 두고 있다. 둘째로 같은 다이어그램이 여러 가지 존재할 때 이를 적절하게 통제해주고 관리해 줄 수 있는 형상관리 기능이 부족하다. 셋째로 다중 사용자 지원 측면에서는 같은 시점에 같은 도형을 여러 사용자가 변경을 하지 못하게 하는 적절한 통제 기능이 부족하다. 넷째로 개발 전 단계 중 분석 또는 설계 등 일부 단계만 지원하고 있어 개발 전 단계를 일관성 있게 개발할 수 있는 환경이 지원되지 못하고 있다. 다섯째로 개발 과정상에 있어서 어떤 한 단계로부터 다음 단계로의 정보의 변환이 원활하게 이루어지지 못하고 있다. 여섯째로 궁극적인 목표인 코드의 자동 생성 수준은 아직 코드의 템플리트 정도를 생성하는 수준에 머물러 있다.

이상과 같이 객체지향 CASE가 갖추어야 할 기능들을 충실히 지원하고 현재 부족한 기능들을 충분히 보완한 객체지향 CASE의 개발을 기대한다.

참 고 문 헌

1. D. Brumbaugh, "Object-Oriented Development : : Building CASE Tools with C++", John Wiely & Son, Inc. 1994.
2. M. Prabandham, W. Selfridge, D. Mann, "The Role of the IRDS", Database Programming and Design, April 1990.
3. "Software Through Pictures/Object Modeling Technique", Preliminary Documentation. Interative Development Environments Inc. 1993.
4. J. Rumbaugh, "Going with the flow : Flow graphs in their various manifestations", Journal of Object-Oriented Programming, June. 1994.
5. Graham, "Object-Oriented Methods", 2nd-ed. Addison-Wesley, Prentice-Hall, 1991.

6. G. Booth, "Object-Oriented Analysis and Design", 2nd ed., The Benjamin/Cummings Publishing Co. 1994.
7. 本位田, "オブジエクト指向 システム開發", 日經BP出版センター.
8. 本位田, 小高, "知的ソフトウエア開發環境を目指す", コンピュータ科學, Vol. 2, No. 2, 1992.
9. 양해술, "객체지향 CASE 환경의 기초 기술", 한국정보과학회지 제9권 제2호, 1991. 4.
10. 양해술, 조영식, 이용근, "객체지향 설계 방법론의 비교 분석", 한국정보과학회지 제11권 제2호, 1993. 4.
11. 양해술, "CASE 개발 기술과 고도 이용", 정보산업 제139~145호, 1993. 11~94. 4.
12. 양해술, "I-CASE의 도구 통합과 방법", 한국정보과학회지 제12권 제2호, 1994. 3.

양 해 술

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)
1878년 성균관대학교 정보처리학과 정보처리 전공(석사)
1991년 日本 오사카대학교 기초공학부 정보공학과 소프트웨어공학 전공(공학박사)
1975년~1979년 육군중앙경리단 전산실 시스템분석장교
1986년~1987년 日本 오사카대학교 객원연구원
1993년~1994년 한국정보과학회 학회지 편집부위원장
1980년~1995. 5 강원대학교 전자계산학과 교수
1994년~1995년 한국정보처리학회 논문지편집위원장
1994년~현재 한국산업표준원(IIS) 이사
1995년~현재 한국소프트웨어품질연구소(INSQ) 소장
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, SA/SD, OOA/OOD/OOP, CASE, SI), 소프트웨어 프로젝트관리

이 응 근

1988년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1994년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1989년~1992년 강원대학교 전자계산학과 조교
1994년~1995년 한림전문대학 전자계산과 강사
1995년~현재 강원대학교 대학원 전자계산학과 박사과정
1996년~현재 경희대학교 전산공학과 강사
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, 객체지향 프로그래밍, 객체지향 분석과 설계 방법, CASE)

이 하 용

1993년 강원대학교 전자계산학과 졸업(이학사)
1995년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1995년~현재 강원대학교 대학원 전자계산학과 박사과정
1996년~현재 경희대학교 전산공학과 강사
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, 객체지향 프로그래밍, 객체지향 분석과 설계, CASE)