特輯 技術解說

# CASE

金 熙 天, 禹 治 水
서울大學校 計算統計學科

## Ⅰ. 서 론

고품질의 대형 소프트웨어 시스템을 높은 생산성과 비용 절감적인 방법으로 개발하고자 하는 것이 1990년대의 기술, 공학, 제조의 모든 분야에서 큰 관심사가 되고 있다. 따라서, 소프트웨어의 생산과 유지에 필요한 인력과 비용의 효율성 증대를 위해 자동화에 근간을 둔 소프트웨어 파라다임에 관한 연구가 필요하게 되었다. 이러한 요구를 충족시키기 위한 것이 CASE(computer aided software engineering)이다.

CASE는 글자 그대로 풀이하면 컴퓨터를 이용한 소프트웨어 공학이다. 소프트웨어 공학이 소프트웨어를 만드는 데 기본적이고 정형화된 구조 또는 메카니즘을 제시하여 소프트웨어의 생산성과 질을 향상시키고자 하는 것이므로, CASE는 소프트웨어 개발 방법의 자동화를 지향하는 시스템으로 소프트웨어 생명 주기의 전체 단계를 연결시켜 주고 자동화시켜 주는 통합된 도구들을 제공하는 것이다.[1]

CASE 도구는 소프트웨어 개발의 정형화와 설계 프레임워크를 제공하는 여러 구조적 방법론의 발전으로 이를 소프트웨어 프로그램으로 구현한 것이다. 즉, CASE는 컴파일러, 디버거, SCCS 등이 지원했던 프로그램의 구현과 유지보수 작업뿐만 아니라 분석과 설계 작업을 자동화 시킴으로써 소프트웨어의 생산성 문제를 해결하고자 한다. CASE 기술은 소프트웨어 생명주기 전체 단계에서 사용되는 개발 도구와 방법론의 결합체이다. 방법론은 구조적 분석, 구조적 설계, 구조적 프로그래밍과 같은 손으로 직접 수행하는 소프트웨어 개발 방법론을 말한다. CASE 기술은 단지 소프트웨어 구현의 해결책에 초점을 맞추는 것이 아니라 전체적인 소프트웨어 생산성과 신뢰성 문제에 초점을 맞춘다는 면에서 전통적인 소프트웨어 기술과 차이를 보이고 있다.

CASE 도구의 상당수는 그래픽 기능을 갖춘 마이크로 컴퓨터나 워크스테이션 상에서 사용자 편의의 인터페이스 기능을 포함한 워크벤치(workbench) 형태로 구성되어 있고 개별적이고 전문적인 소프트웨어 개발자의 생산성 향상에 초점을 맞추고 있다는 점에서 단지 최종 사용자가 주로 사용하는 4세대 도구들의 개선 정도로 생각해서는 안된다.

## Ⅱ. 현 소프트웨어 개발상의 문제와 CASE의 영향

소프트웨어 개발자들은 개인용 컴퓨터에서 메인 프레임급에 걸친 모든 하드웨어 상에서 좀 더 지적이고 강력한 소프트웨어를 개발하여 최종 사용자에게 공급하고자 노력하고 있다. 그러나, 소프트웨어 프로젝트가 점차로 대형화되고 복잡해짐에 따라 이러한 작업들은 종종 예기치 못한 결과나 실패를 초래하게 된다.

한 연구에 의하면 소프트웨어 에러의 64%가 소프트웨어 생명 주기상의 요구분석 단계와 설계 단계에서 야기되었으며 이중 30%만이 인수 검사(acceptance test) 전에 발견되었다고 보고하고 있다.[2] 이는 소프트웨어 개발시의 분석과 설계 작업의 중요성을 시사하고 있다.

CASE의 본질은 바로 소프트웨어 생명 주기 후반단계에서의 노력과 비용을 줄이기 위해 대형 소프트웨어 프로젝트의 설계와 개발상의 고질적인 문제를 줄이거나 제거하는 것이다.

CASE 시스템이 사용자에게 제공하는 이점으로는 소프트웨어 공학 개념의 적용, 자동화된 검사를 통한 소프트웨어의 품질 향상, 프로그램 유지보수의 간편, 개발 기간

의 단축 및 비용 절감등을 들 수 있다.

CASE 환경이 생산성과 프로젝트 비용과 소프트웨어의 품질에 어떠한 영향을 미치는가를 조사하기 위해 CASE 환경 EPOS를 22개 프로젝트에 적용해 본 결과 프로젝트를 용이하게 수행할 수 있었으며 문서가 대부분 자동 생성되었고 소프트웨어의 품질이 향상되었으며 유지보수 비용의 큰 절감을 기할 수 있었다고 보고하고 있다.[3]

## Ⅲ. CASE 시스템의 구성 요소

CASE 시스템은 지원 범위에 따라 다음과 같이 분류할 수 있다.[1]

- ○ CASE 도구함(toolkit)
  소프트웨어 생명 주기의 한 단계나 특정 소프트웨어 작업 분야를 자동화시키기 위한 도구의 집합으로, 분석 및 설계 방법론을 지원하는 것을 상위 CASE로, 구현과 유지 보수 단계를 지원하는 것을 하위 CASE라 한다.
- ○ CASE 워크벤치(workbench)
  전체 소프트웨어 생명 주기 작업들을 자동화 시켜주기 위한 통합된 도구의 집합
- ○ CASE 방법론 동반기(methodology companion)
  특정 소프트웨어 개발 방법론의 작업들을 단계별로 지원하거나 어느 한 단계의 모든 개발 방법들을 지원하는 도구의 집합

CASE의 성장은 마이크로 컴퓨터와 워크스테이션의 발전과 보급에 힘입었으며 CASE 도구는 이러한 하드웨어와 LAN(local area network)을 중심으로한 하드웨어 platform에 탑재되어 소프트웨어 개발 환경을 구성하며 CASE 워크벤치는 소프트웨어 개발 작업과 관리까지 지원하도록 다음과 같은 기능을 갖추어야 한다.[1]

- ○ 명료한 사고와 문서 생성을 위한 그래픽 기능
- ○ 비용 절감을 위한 자동 오류 검사
- ○ 전체 개발 과정에 필요한 정보 관리와 도구의 통합을 위한 정보 저장소
- ○ 순공학과 역공학을 통한 소프트웨어 생명 주기의 통합 및 지원
- ○ 프로토타이핑의 지원
- ○ 구조적 방법론의 지원
- ○ 자동 코드 생성

### 1. CASE 소프트웨어 개발 환경

CASE 도구는 생명 주기의 전반부 즉, 타당성 조사, 요구 분석, 시스템 설계 뿐만 아니라 프로젝트 관리, 유지보수를 포함한 광범위한 분야의 작업을 자동화하며 CASE 소프트웨어 개발 환경은 이러한 것을 지원하는 다양한 도구들로 구성된다. 현재 상용화 되어있는 CASE 도구들은 주로 구조적 다이어그램의 작성과 시스템 문서의 생성을 자동화하며 양질의 소프트웨어 개발에 필요한 시스템 명세에 대한 완전성과 일치성을 검사하는 기능을 가진다.

그림 1에 CASE 환경의 구성 요소를 나타내었으며 아직까지 이러한 도구를 모두 포함하는 시스템은 거의 없다.

이 그림은 기술적으로 높은 수준에 있는 CASE 시스템의 핵심 구성 요소를 나타내고 있으며 각 도구간의 인터페이스가 존재하고 또한 자연스러워야 한다. CASE 도구의 부가가치를 높일 수 있는 방법의 하나가 도구들을 통합하여 사용하는 것이다. CASE 소프트웨어 개발 환경은 단순한 도구의 집합이 아닌 고수준으로 통합된 도구들의 집합이 될 것이며 이 도구들은 사용하기 쉽도록 통일성이 있어야 하며 함께 존재할 뿐만 아니라 다른 도구를 호출하고 다른 도구에 자동적으로 자료를 전송하는 기능을 가져야 한다.[4]

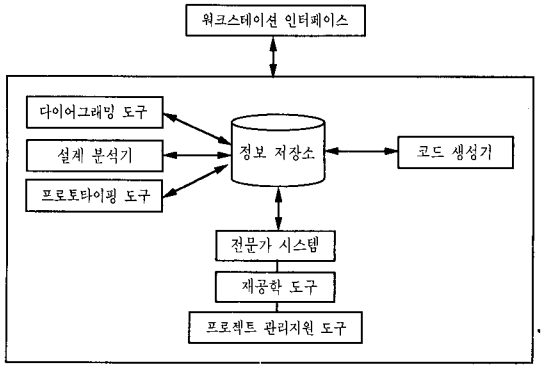


**그림 1.** CASE 시스템의 구성요소

1) 다이어그래밍 도구

다이어그램은 소프트웨어 개발에 있어서 항상 중요한 역할을 담당하여 왔으며 소프트웨어의 명세서를 정의하고 설계를 표현하는데 사용되며 코드로 구현하기 위한 청사진을 제공한다. 대부분의 분석, 설계 기법에서 다른 개발자 또는 사용자와의 의사 전달 수단으로 그래픽 표현을 이용하나 다이어그램을 손으로 그리는 것은 매우 지루하

며 시스템에 변화가 생기면 다이어그램을 다시 그려야하는등 많은 양의 작업을 필요로 하게 된다.

CASE 도구들은 정형화된 그래픽 표현 기법을 이용하고, 정보 저장소로부터 필요한 정보를 얻어서 그래픽 편집기를 사용하여 자료 흐름도, 구조도, 객체 관계도 등 다이어그램 생성의 자동화를 지원한다. CASE 시스템이 본격적으로 사용된 것도 마이크로 컴퓨터가 보편화 되기 시작한 후 구조적 기법을 지원하는 다이어그래밍 도구가 보급되면서 부터이다.

2) 설계 분석기

자동적인 오류 검사는 소프트웨어 생명 주기 과정중 초반에 오류를 발견할 수 있도록 도와주어 개발 비용을 감소시킨다. 다이어그래밍 기법이 기반을 두고 있는 구조적 접근 방법은 특정 규칙들을 지니고 있다. 예를들어, 자료 흐름도에서 프로세스는 입출력 모두를 가지고 있어야 한다.

양질의 소프트웨어 개발에 중요한 역할을 하는 것이 소프트웨어 명세서에 대한 오류를 검사하고 분석하는 것이고 분석 결과가 자동적으로 제공되어야 한다. 설계 분석기의 기능은 이와 같은 완전성, 일치성등을 검사하는 것이며 정형적인 정보 모델에 기반을 두어야 한다. 또한 분석이 완료된 명세에 대해 현재의 가능한 설계 대안을 비교, 평가할 수 있어야 하며 현재의 CASE 도구들은 극히 한정된 능력을 제공한다. 그러나, 현재 설계 분석기는 효율적인 편집기, 지능형 전문가 시스템을 포함하는 방향으로 개선되고 있다.[5]

3) 코드 생성기

코드 생성기는 프로그램 설계 명세서로 부터 실행 가능한 코드를 생성함으로써 구현 단계를 자동화 시킨다. 현 기술 수준으로는 코드가 골격(skeleton) 프로그램인 것이 보통이며 제한된 응용 범위에서는 완전한 코드 산출도 가능하다. 전자의 경우에는 데이타베이스, 화일, 화면과 보고서 서술에 관한 코드와 주석을 포함한 개요 형태의 절차적인 논리가 자동 생성되며 추가적인 프로그램 논리는 손으로 작성한다. 완전한 코드 생성을 위해서는 전반부 CASE 도구들이 생성했던 정보 저장소로 부터의 정보 이외에 화면 정의, 보고서 정의, 대화 및 문서 정의 등의 자세한 사항들이 필요하다.

4) 정보 저장소

오랜 기간 동안에 걸쳐서 개발되는 대규모 소프트웨어의 개발 과정에는 여러가지 종류의 정보와 그에 따르는 막대한 양의 정보가 발생하며 그 자료 항목들은 여러가지 형태로 저장되어 유지된다. 프로그래밍 언어로 작성된 원시 코드, 작동되는 기계에 의존하는 목적 코드, 요구사항이나 분석 및 설계 명세서와 같은 문서 양식, 그리고 개발 팀원간에 상호 교환된 메모나 전자 우편 같은 자료들이 소프트웨어 프로젝트 수행중에 발생할 수 있는 자료의 형태들이라고 할 수 있다. 정보 저장소는 이들 자료간의 상관관계 분석이나 타당성 검사 등의 기능이 포함될 수 있으며 그래픽 정보와 시스템 설계 정보등을 포함하고 또한, 단순히 관련된 정보의 집합이 아니기 때문에 자료사전이나 데이타베이스와 달리 정보저장소라 불린다.

소프트웨어 개발 생산성을 향상시키는 중요한 방법중 하나가 개발자에게 필요할 때 유용한 정보를 제시해 주는 것이라고 할 때 CASE 정보 저장소는 CASE 시스템의 핵심이라 할 수 있다. CASE 정보 저장소는 소프트웨어 개발 과정에서 나타난 정보를 저장, 엑세스, 변경 및 분석하고 보고서를 생성하는 기능을 수행하며, CASE 도구들은 작업을 수행할 때 이 정보를 공유하면서 개발하고자 하는 시스템에 관한 정보를 생성하고 관리하는 기본적인 기능을 가지고 있다.

CASE 정보 저장소는 다음과 같은 기능을 뒷받침한다.[5,6]

- CASE 도구들의 상호 연결 및 통합
- 시스템 명세서의 일치성 및 무결성 제어
- 시스템 정보의 공유
- 문서의 표준화
- 시스템 문서 생성 및 프로그램 코드의 자동 생성
- 프로젝트 관리 및 제어
- 정보의 재사용

5) 전문가 시스템

인공지능 분야의 지식 표현 및 활용 기법들을 소프트웨어 개발 및 관리에 이용하면서 인공 지능 도구들이 활용되고 있다. 소수의 도구는 현재 전문가 시스템을 대상 시스템의 분석 및 설계에 이용하여 일관성 있고 정확한 명세의 생성을 위해 필요한 분석과 사용자 안내를 위한 지식을 제공한다. 또 역공학과 같이 많은 결정이 불충분한 지식에 기반을 두고 이루어지는 분야에 유용하다.

몇몇 CASE 시스템은 아래의 영역에 전문가 시스템 기술을 이용하고 있다.

TI(Texas Instruments)의 IEF(Information Engineering Facility)와 KnowledgeWare의 IEW(information engineering workbench)는 전문가 시스템이 방법론을 이해하고 있어서 방법론의 정확한 수행에 도움을 준다. Cortex의 Vision은 대화 방식을 이용하여 생명주기 과정을 지원하며 Transform Logic Corporation의 Transform은 구조적 코드의 규칙과 재사용 가능한 구조를 이해하여 코드 생성을 돕는다.

6) 프로젝트 관리 지원 도구

자동화된 프로젝트 관리 도구는 프로젝트 관리자로 하여금 소프트웨어 프로젝트에 관한 진행 사항의 추적과 보고, 자원 관리의 제어등으로 작업 능력을 향상시키도록 도와주며, 효율을 극대화하기 위해서는 프로젝트 관리 지원 도구가 모든 CASE 정보 저장소를 엑세스하여 모든 시스템 정보를 얻을 수 있어야 한다.

프로젝트 관리 분야를 다음과 같이 세분할 수 있다.[7]

- 프로젝트 계획 및 예측
- 시스템 개발 방법론과 표준에 대한 온라인 엑세스
- 프로젝트 문서 제어 및 버전 제어
- 사건 기록, 시간 회계, 상태 보고
- 문제 기록 및 추적

이러한 프로젝트 관리 지원 환경을 IPSE(integrated project support environment)라 하며 Softlab의 MAESTRO와 AGS의 MULT/CAM이 있다. MAESTRO는 각 프로그램에 발생되는 모든 변경사항을 로그가 유지함으로써 초기 버젼을 복원할 수 있으며 데이타 베이스를 이용하여 문제 추적 및 해결에 도움을 준다.

7) 프로토타이핑 도구

프로토타이핑 도구는 사용자에게 시스템의 요구 사항과 시스템의 작동을 신속하게 제시하여 생명 주기 전반부의 작업들을 자동화하며 화면과 보고서 양식의 설계 작업을 돕기도 한다. 또한 실제 개발될 시스템의 인터페이스를 구축하는데 이용된다.

명세서와 재사용 가능한 소프트웨어 부품을 통한 프로토타이핑으로 인해 프로그래밍 생산성과 소프트웨어 신뢰도 향상등 소프트웨어 기술의 진일보를 이룩하였으며, 오늘날 고수준의 명세서로 부터 자동 코드 생성은 실용화되지 않고 있으나 프로토타입의 자동 생성은 가능하다.

8) 재공학 도구

기존의 코드가 구조화 되어있지 않고 각 시스템마다 중복된 데이타나 누락된 데이타가 있으면 유지보수가 힘들어지고 자원의 비효율성을 초래한다. 재공학 도구는 기존의 코드를 구조적으로 재구성해 주는 도구와 역공학(reverse engineering) 도구가 있다. 역공학 도구는 기존 시스템에 대한 구조도, 자료 사전같은 설계 명세서를 생성해 주는 도구이다. 이는 기존 시스템의 정보를 CASE 정보 저장소에 보관할 수 있게 함으로써 기존 시스템과 새로운 시스템을 완전하게 통합시켜 주는 기능을 제공하며 개발된 시스템의 유지보수 작업에 있어, 이의 효율적인 관리를 위해 문서화를 하는 경우 발생될 수 있는 여러 문제점을 해결할 수 있다.

## 2. CASE 하드웨어의 구성

그림 2에 나타낸 것과 같이 CASE 소프트웨어 개발 환경에 대한 세가지 하드웨어 구조 형태가 있을 수 있다.[8] 이 중 가장 간단한 형태가 독립된 워크스테이션 구조로 하드웨어 구성은 HP9000, SUN, Apollo, IBM RT PC와 같은 워크스테이션이나 개인용 컴퓨터로 이루어진다. Index Technology의 Excelerator가 이러한 예이다. 그러나, 실제적으로는 CASE 워크스테이션은 독립적으로 존재할 수 없으며 프로젝트 팀 구성원 간에 정보를 공유하고 도구간에 자료들을 교환할 수 있도록 다른 개발자의 워크스테이션이나 다른 하드웨어 시스템에 연결하는 것이 필요하다. 이 때 워크스테이션의 연결은 LAN에 의해 이루어지게 되며 하나의 워크스테이션이 CASE 정보 저장소로 이용된다.

2단계 구조에서는 정보 저장소가 메인 프레임이나 고성능 미니 컴퓨터에 존재하며 이와 연결된 워크스테이션에서 다이어그래밍과 같은 분석과 설계 작업을 하며, 코드 생성 및 분석과 테스팅 같은 작업은 메인 프레임에서

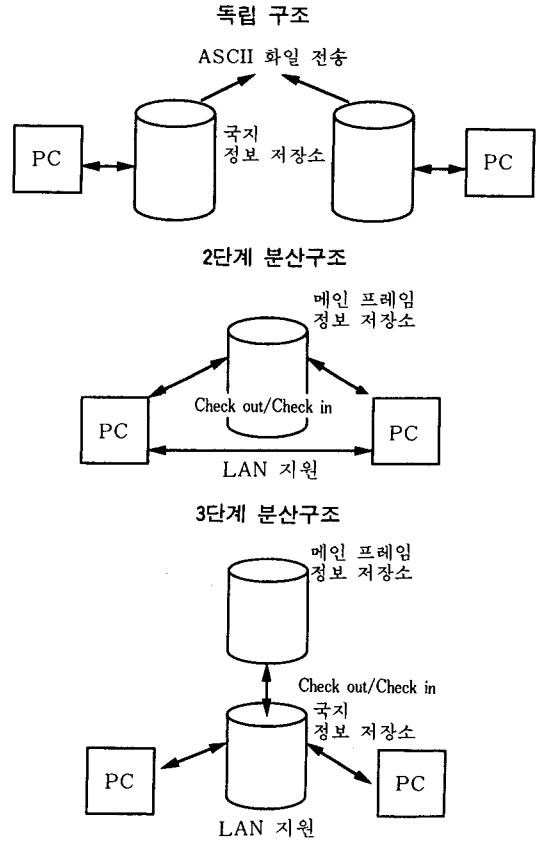


**그림 2.** CASE의 구조

| 제 품 명 | 개 발 회 사 | 생명주기 지원정도 | 사 용 환 경 | 방 법 론 | 특 징 |
|---|---|---|---|---|---|
| IEW | KnowledgeWare | 전 체 | IBM PC/AT, OS/2 | Information Eng. Yourdon/Demarco Gane/Sarson | 4가지 모듈(Planning, Analysis, Design, Construction)로 된 하나의 도구 집합. 다이어그램 도구, PC저장소, 설계분석기, 코드생성기, 전문가 시스템 규칙으로 구성됨. 개방형 아키텍처 지원. 기존 시스템의 재개발 및 재사용 가능. |
| IEF | Texas Instruments | 전 체 | IBM PC, OS/2 IBM MVS | Information Eng. | 4가지 도구 집합(Planning, Analysis, Design 도구 집합은 워크스테이션 상에서 코드 생성, 다이어그램 테스팅, DB 생성으로 구성된 Construction 도구는 OS/2와 MVS에서 작동). 분산 환경 지원. |
| Foundation | Arthur Andersen | 전 체 | DEC, IBM, Bull | 자체 방법론 | 여러 H/W와 O/S에서 작동되는 통합 개발 도구의 집합으로 아래 모듈로 구성. METHOD/1은 자체 개발된 PC 기반의 방법론으로 프로젝트 관리 기능. DESIGN/1은 시스템 설계부분을 자동화하고 LAN을 지원. INSTALL/1 은 구현 유지 단계를 지원하는 메인프레임 환경. PLAN/1은 다이어그램 편집기를 통해 정보 계획 지원. |
| TeamWork | Carde Tech. | 상 위 | Apollo, HP IBM PC, Sun | Yourdon/Demarco | 시스템 분석 및 설계 기능이 뛰어남. Teamwork/SA, Teamwork/RT( 실시간 분석 모듈), Teamwork/IM, Teamwork/SD로 구성. 다양한 문서 생성과 역공학 지원 기능. 공학 분야에 널리 쓰이고 있음. |
| Excelerator | Index Tech. | 상 위 | IBM PC | Yourdon/Demarco G/S. Jackson | 구조적 분석 및 설계, 다이어그램 작성. CASE 도구의 시초. 사무 분야에 널리 쓰임. |
| Softbench | HP | 하 위 | HP, Apollo | 구조적 방법론 객체지향 방법론 | 분산 컴퓨팅 지원. 온라인 도움말 |
| VS-Designer | Visual Software | 상위 | | Y/D, Warnier-Orr Jackson, Info Eng | 방법론의 적응성과 정제(customization) 가능성. |
| Corvision | Cortex | 전 체 | IBM PC, DEC VAX VAXstation | Y/D, Info Eng. | 사용자의 요구로 부터 응용프로그램의 논리적 흐름을 시각적으로 표현하여 재구성하는 Menu Diagrammer 존재. |
| Software through Pictures | Interactive Development Environments | 상 위 | Apllo, DEC, HP Sun | Y/D, Yourdon/Con-stantione, E-R, Jackson | 각종 다이어그램을 지원하는 그래픽 편집기. 개방형 아키텍처 추구, 도구간의 통합 강조(모든 도구가 잘 정의된 입출력 ASC-Ⅱ화일 형태를 가짐). |
| TELLON | Pansophic | 전 체 | IBM PC | | 강력한 프로토타이핑 기능. 프로그램 구조화 기능. |

**그림 3.** CASE 도구의 비교

수행한다. IEF등이 이에 해당된다. 어떤 CASE 시스템은 호스트 컴퓨터와 워크스테이션 사이에 중간 하드웨어를 두어 프로젝트 팀을 지원하는 화일 서버의 역할을 담당하게 하여 메인 프레임의 자원에 대한 집중을 감소시키기도 한다. 이와 같은 구조는 유럽에서 보편화된 구조다.[10]

## Ⅳ. CASE의 사용 현황 및 전망

CASE 도구의 역사는 1977년에 미시간 대학에서 개발된 요구사항 명세와 분석을 위한 PSL/PSA에서 기원을 따질 수 있으며 80년대 중반부터 문서화 도구와 다이어그래밍 도구로 사용되어지면서 알려지기 시작하였는데 설계 자동화 특히, 구조적 설계의 다이어그램 작성을 지원하였다. 실질적인 CASE 도구의 효시는 Index Technology의 Excelerator로 정보 시스템과 실시간 시스템을 설계하고 문서화하는데 이용되며 개인용 컴퓨터에 기반을 두고 있다. 이 도구의 사용자들은 요구 분석 및 설계 단계에서 평균 30～40%의 생산성 향상을 기할 수 있었으며 명세서의 오류와 불일치를 발견하여 수정함으로써 고품질의 시스템을 개발할 수 있었다고 보고하고 있다.[9] 80년대 후반에는 설계 자동화와 구현 자동화를 연결시키는 것이 주요 과제였다. 그림 3에 상용화된 몇 가지 CASE 도구를 비교하였다.[5,7,8]

대규모 소프트웨어에 대한 수요가 증가함에 따라 효율적인 개발도구의 필요성은 점차 증가하고 있으며 CASE는 초기 요구 분석 단계부터 시스템의 설치와 유지 보수 단계에 이르는 전체 소프트웨어 생산 과정의 지원을 원하는 요구에 부응하여야 한다.[11]

CASE 도구는 워크스테이션의 성능 향상 및 보급, 사용자 인터페이스의 개선, 생산성 향상으로 인한 인식 제고등으로 계속 수요가 상승하고 있으며 소프트웨어 개발을 위한 친숙하고 보편적인 도구가 되기 위해서는 다음과 같은 사항들이 개선되어야 한다.

첫째, 개발 과정상의 서로 다른 단계와 서로 다른 성질의 소프트웨어를 지원하기 위해 독립적으로 개발된 도구 간의 통합(integration)이 이루어져야 한다. 이는 공통 정보 저장소의 이용, 정보 저장소를 통한 정보의 제공, 사용자 인터페이스의 통일성등의 문제를 가지고 있다.

둘째, 서로 다른 종류의 소프트웨어는 서로 다른 방식으로 만들어지게 되며 생산 환경이 서로 다른 사용자는 자기에 맞는 시스템을 필요로 하게되므로 특정 하드웨어나 운영체제에 의존하지 않으며 사용상의 호환이 가능하도록 적당한 정제 기술(customization technique)이 필요하다.

셋째, 정보 저장소는 소프트웨어 개발 환경의 핵심 요소이다. 따라서 정보 저장소에 대한 표준 사용자 인터페이스가 필요하며 이는 계속해서 표준 객체 관리 기능을 가지는 정보 저장소의 개발이 필요하다.

넷째, 많은 개발자에 의한 대형 프로젝트는 적절한 도구의 지원없이는 관리할 수 없으므로 전체 개발 과정을 통제할 정형화된 과정 모델이 필요하다.

미래의 CASE 도구는 분석, 설계와 같은 소프트웨어 개발 과정상의 한 단계 또는 두 단계에서 특정 방법론을 기준으로 운용되는 것에 그치지 않고 보다 통합된 기능으로 개발 과정상의 전체 단계 자동화를 달성하는 방향으로 발전할 것이다. 통합 CASE의 경우 공통 저장장소, 공통 인터페이스외에 소프트웨어 개발 단계의 모형 제시와 개발에 이용되는 모든 도구들이 이용할 수 있는 표준 CASE 정보 저장소가 필요하다. 근본적으로는 설계 명세서에서 자동적으로 소프트웨어를 생성시킬 수 있어야 한다. 인공지능 분야 특히 전문가 시스템의 응용 폭이 늘어남에 따라 프로그램 명세에서 지식을 기반으로 한 효율적인 프로그램의 생성을 지원할 수 있으며 프로토타이핑 방법과 사전에 구축된 재사용 가능한 소프트웨어 부품에 관한 지식이 활용될 것이다. 코드 생성 문제는 현재 CASE 기술에서 하나의 장벽이나, CASE 기술이 발전하고 도구의 응용 범위가 확장하고 표준화에 대한 연구가 진행됨에 따라 조금씩 허물어 질 것이다.

한편 최근 국내 기업들도 효과적인 소프트웨어 업무 추진을 위해 CASE의 도입을 추진하여 IEF, IEW, 앤더슨컨설팅의 FOUNDATION, HP의 Softbench, Excelerator 등이 국내에 보급되었다. 아직은 CASE의 활용이 초보 단계에 불과하나 4GL의 기능이 CASE 기능에 통합됨에 따라서, 그리고 방법론의 부재, 고급 인력 부족, 한글화 문제등이 해결되면 CASE의 활용이 본격화될 것으로 보인다.

## Ⅴ. 결 론

소프트웨어 인력과 기술의 한계에 기인한 소프트웨어 위기를 극복하는 근본적인 해결책으로 제시되고 있는 것이 CASE이다. 또한 앞으로 CASE가 보편적인 도구로 계속 성장하기 위해서는 사용자 편의성을 높이기 위해 지

능적이고 통합적인 $I^2$ CASE(intelligent & integrated CASE)로 발전해야 한다.

그러나, CASE가 소프트웨어 개발을 도와주는 것이지 소프트웨어를 자동으로 생성하는 것이 아니기 때문에 CASE에 대한 지나친 과신을 하여서는 안되며 조직내 고유의 개발 방법론이 없는 상태에서 성급히 CASE를 도입하면 위험한 결과를 초래할 수 있으므로, 충분한 타당성 연구와 교육, 전문가와의 상담을 통해 도입해야 할 것이다.

## 參 考 文 獻


[1] C. McClure, *CASE is Software Automation,* Prentice-Hall, 1989.

[2] W. Suydam, "CASE Makes Strides Toward Automated Software Development," *Computer Design,* pp. 29-70, January 1987.

[3] P. Lempp and R. Lauber, "What Productivity Increases to Expect from a CASE Environment: Result of a User Survey," *Productivity: Progress, Prospects, and Payoff,* pp. 13-19, June 1988.

[4] D. L. Burkhard, "Implementing CASE Tools," Journal of Systems Managements, pp. 20-25, May 1980.

[5] P. A. Ng and R. T. Yeh, *Modern Software Engineering,* Van Nostrand Reinhold, 1990.

[6] P. B. Henerson and D. Notkin, "Integrated design and programming environments," *IEEE Computer,* pp. 12-16, November 1987.

[7] C. Gane, *Computer-Aided Software Engineering, the Methodologies, the Products, and the Future,* Prentice-Hall, 1990.

[8] *CASE the Potential and the Pitfalls,* QED Information Sciences, Inc., 1989.

[9] E. J. Chikofsky and B. L. Rubenstein, "CASE: reliability engineering for information systems," *IEEE Computer,* Mar. 1988.

[10] 이형원, 우치수, "CASE 시스템의 구성요소," 정보과학회지, 제9권 제2호, pp. 21-27, 1991년 4월.

[11] 우치수, 소프트웨어工學, 三星尖端技術硏修所, pp. 6-1-6-28, 1990년 1월.


## 筆 者 紹 介


**金 熙 天**

1966年 2月 20日生

1989年 2月 서울대학교 계산통계학과 졸업

1991年 2月 서울대학교 계산통계학과(석사), 전산과학 전공

1991年 3月 ~ 현재 서울대학교 계산통계학과(박사과정)

주관심 분야: CASE, 소프트웨어 개발 환경 등



**禹 治 水**

1946年 11月 6日生

1972年 2月 서울대학교 응용수학과 졸업

1977年 2月 서울대학교 계산통계학과(석사)

1982年 2月 서울대학교 계산통계학과(박사)

1972年 3月~1974年 1月 과학기술연구소 연구원

1978年 3月~1978年 8月 영국 라퍼리대학 연구원

1975年 12月~1982年 8月 울산공대 전산학과 조교, 부교수

1985年 1月~1986年 1月 미국 미시간대학교 postdoc.

1982年 3月~현재 서울대학교 계산통계학과 조교수, 부교수

주관심 분야: 소프트웨어 공학, 프로그래밍 언어 등