□ 기술보고 □

통합 CASE 구현 환경 하에서의 '들무새' 시스템

시스템공학연구소 이승지* · 신규상* · 이단형

목 차

1. 서 론

소프트웨어 개발의 생산성 및 품질향상의 궁극적인 목적을 달성하기 위해 개발에 영향을 줄 수 있는 모든 요소들을 고려한 종합적인 접근방법이 개발환경을 중심으로 계속 연구되고 있다. 특히, 자동화도구(Automated Tool)는 개발자의 생산성을 향상시키며, 개발자의 기술적 필요성은 신규 자동화 도구의 개발/도입 및 새로운 컴퓨터 시스템의 도입을 촉진시킨다. 즉, 직면하고 있는 소프트웨어 위기를 극복하고 개발 생산성 및 품질 향상을 제고하기 위한 해결책의 일환으로 CASE 도구에 대한 인식이 확산되고 있다. 그러나 우리의 현황은 국내 개발 또는 CASE의 활용 사례가 많지 않아 도입의 필요성은 느끼고 있으나 성공적인 적용에 대한 확신이 결여 되어 있는 등 아직은 도입단계로서 선진국의 기술개발 내용을 상당히 받아들여야 하는 입장으로서는 우리 실정에 적합한 기술 및 방법을 정립하고, 나아가 사용자에의 보편적 활용이 될 수 있도록 토양을 마련하는 역할이 중요하다고 볼 수 있다.

본고에서는 이러한 관점에서 시스템공학 연구소에서 개발하고 (주)유니온 시스템과 (주)현영 시스템즈에서 최근 상용화한 4개 도구의 기능에 대해 고찰하고 이어서 93년 10월부터 착수된 '들무새' 시스템의 개발방향에 대해 논하고자 한다.

2. SERI 개발도구 현황

'90년 7월부터 3년간에 걸쳐 개발한 단계별 지원 CASE도구는 START, STADE, COSTAR, COMET 등 모두 4개의 모듈로 구성되어 있으며 현재, 참여업체와의 실시 계약에 의해 상품화 보급중이다. UNIX O.S.하 X Window 환경에서 한글을 지원하며 분석 및 설계단계를 지원하는 시스템과, 개발과정에서의 S/W 형상 및 버전을 관리하고 소스 프로그램에 대한 품질 측면에서의 테스트 및 문서화를 가능케하는 등의 기능을 수행토록 되어 있다.

2.1 구조적 분석지원도구 (START)

START(STructured Analysis and Reporting Tool) 시스템은 현재 요구분석 기법으로 가장

* 정회원

널리 사용되고 있는 구조적 분석 기법 대상으로 그 중에서도 DeMarco, Yourdon의 표시법을 지원한다.

구조적 분석 기법에서 사용되는 객체형은 외부실체,프로세스, 자료흐름, 및 자료저장소로 정의할 수 있다. START에서도 이와 동일하지만, 자료사전과 속성 편집을 위하여 자료흐름 객체형을 자료흐름 구성관계에 따라 이를 자료그룹(group)과 자료원소(element)로 세분하였다. START 시스템의 전반적인 구성은 그림 1과 같다.

사용자는 명세편집기를 통하여 자료흐름도, 자료사전, 속성 및 기능명세 편집 작업을 수행하고, 기능명세를 제외한 각 편집내역에 대한 분석을 명세분석기를 통하여 수행한다. 또한 사용자는 보고서 산출기를 통하여 데이타베이스에 수록된 내용을 다양한 형태의 보고서로 조회할 수 있으며 데이타베이스 관리기를 이용하여 직접 데이타베이스 내용을 수정할 수도 있다.

시스템 관리기는 모든 작업에 선행하여 사용환경을 정의하며 작업 대상 시스템에 대한 선택, 생성, 삭제, 복사, 저장, 및 재생 등의 기능과 사용중인 프린터 기종을 선정하는 기능이 있다.

명세편집기는 요구명세를 편집하는 도구로서 자료흐름도(DFD), 자료사전(DD), 기능명세(MS) 및 속성(PR) 편집기가 있으며 자료흐름도 편집기는 문법지향적 편집기(syntax-directed editor)로서 오류예방 기능을 제공한다. 즉 사용자 입력사항에 대하여 즉각적인 응답을 보내면서, 편집시 입력 내용이 작성규칙에 적합한지를 검증하고 잘못된 입력에 대해서는 오류 메세지를 출력하고 원래의 상태로 복귀함으로써 사전에 오류를 예방하여 시스템 명세의 정확성을 높일 수 있다. 자료사전 편집기는 자료흐름도상에 나타난 모든 자료형태, 즉 자료흐름이나 자료저장소에 대한 구성관계를 자료사전의 작성규칙에 따라 편집하기 위한 것이다. 기능명세 편집기는 자료흐름도에 나타난 모든 프리미티브 프로세스에 대하여 처리논리를 편집하기 위한 것이다. 명세분석기는 명세편집기를 활용하여 작성된 대상시스템에 대한 요구명세 내역이 구조적 기법상의 각종 규칙과 일치하는가를 검증하고 오류가 없

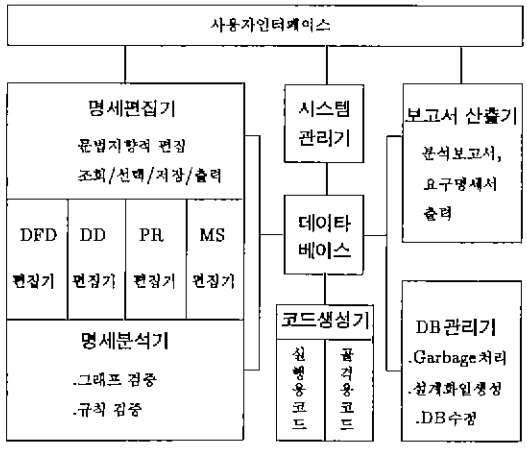


그림 1 START 시스템의 구성도

으면 이를 데이타베이스에 수록한다. 명세분석기에는 자료흐름도 분석기, 자료사전 분석기 및 속성 분석기가 있으며 기능명세서는 일정한 형식이 없이 자유롭게 작성되기 때문에 분석기가 제공되지 않는다. 자료흐름도 분석기는 작성된 자료흐름도 형태에 대한 그래프 검증과 상하위 자료흐름도간의 균형검증을 수행한다. 자료사전 분석기는 작성된 자료정의에 대하여 어휘분석, 구문분석, 형분석, 및 관계분석을 수행한다. 자료사전에 나타난 자료원소들의 형과 그 값을 편집하기 위해서 속성편집기가 제공되며 속성 분석기는 작성된 속성 정의에 대하여 중복여부 및 객체형 분석을 행한다.

보고서 산출기는 데이타베이스에 수록되어 있는 요구명세 내용을 토대로 구조적 명세서를 제공한다. 산출되는 보고서의 종류에는 자료흐름도보고서, 자료사전보고서, 기능명세서보고서, 프로세스구조보고서, 자료저장소 구조보고서, 공통특성보고서, 프로세스보고서를 포함하여 18개의 보고서를 제공하는데 대상 시스템에 대한 이해를 향상시키고 여러각도에서 시스템을 신속하게 파악할 수 있도록 하기위하여 다른 어떠한 종류의 구조적 요구분석을 위한 도구보다 보고서의 종류 및 기능을 많이 제공한다.

데이타베이스는 정보의 저장 및 검색을 원활히 할 수 있도록 정규화 되어 있으며 데이타베이스 관리기는 데이타베이스 항목의 이름변경, 불필요 데이타 제거, 및 설계접속 화일 생성 기능이 있다.

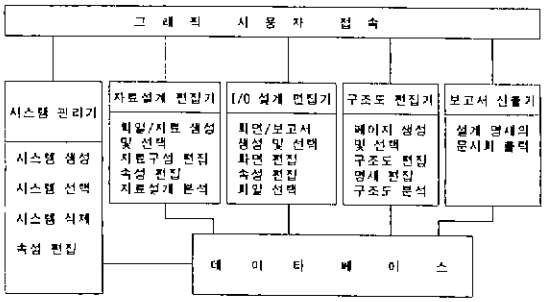


그림 2 STADE 시스템의 구성도

코드생성기는 데이타베이스에 저장된 자료흐름도와 자료사전의 데이타를 기본으로하여 실행을 위한 C 코드와 주어진 응용프로그램에 대한 프로그램의 골격을 자동적으로 생성하여 준다.

2.2 구조적 설계지원도구(STADE)

STADE 시스템은 소프트웨어 개발자가 소프트웨어 개발시의 설계 업무를 효율적이고 용이하게 수행하도록 도와주는 소프트웨어 설계 지원 도구이다. STADE 시스템은 현재 가장 널리 사용되고 있는 구조적 설계(Structured Design) 기법을 지원한다. 구조적 설계 기법은 시스템을 단순.체계화시켜 쉽고 빠르고 정확하게 개발하기 위한 고려사항 및 기법의 집합으로써, 외부설계(External Design)단계 일반설계(General Design) 단계, 상세 설계(Detailed Design) 단계로 구분하여 적용된다.

STADE 시스템은 외부설계 단계에서는 자료설계와 화면 설계를, 일반설계 단계에서는 구조도 설계를, 상세 설계 단계에서는 모듈의 논리설계 등의 설계 업무를 지원한다. 시스템은 소프트웨어 개발자로 하여금 윈도우를 통하여 시스템과 대화형식으로 설계 명세를 편집하게 함으로써, 구조적 설계기법을 수작업으로 수행할 경우의 여러가지 문제점을 제거하여 주며 또한, 설계 명세에 대한 오류를 검증하여 분석보고서를 제공하여 설계 오류를 시스템 개발 초기에 정정할 수 있고, 설계 명세에 대한 각종 양식의 보고서를 소프트웨어 개발자 및 관리자에게 보여줌으로써 소프트웨어 개발의 생산성을 향상시킨다.

STADE의 구성 기능중, 시스템 관리기는 소프트웨어 설계의 모든 작업에 앞서 대상 시스템에 대한 작업 환경을 설정하는 도구이다. 대상 시스템의 선택, 새로운 시스템의 생성, 시스템의 삭제, 시스템의 변경 또는 복사, 디스켓 또는 테이프 등의 보관 장치로의 보관이나 재생, 대상 시스템의 관리자나 설명 등을 입력 변경하는 시스템 관리기능과 프린터 설정 등 시스템 관리 및 환경 설정 등에 관한 사항을 수행한다.

자료설계 편집기는 시스템에서 사용되는 화일, 자료그룹, 자료항목들의 설계 명세를 작성하는 도구 이다. 요구분석 단계의 지원 도구인 START에서 작성된 자료들, 즉 자료 저장소, 자료그룹, 자료항목 등의 자료 개체정보를 이용하여 설계시 사용되는 자료의 형태로 명세화할 수 있도록 하였다.

입출력 명세란 설계자의 설계 활동중 외부설계의 한 부분인 입출력 설계의 산출물을 말한다. 입출력 편집기는 이러한 설계자의 입출력 설계 활동을 지원하고, 설계의 산출물인 입출력 명세를 설계 명세 화일로 담아 두었다가 설계 명세의 일관성 분석 및 완전성 분석에 사용이 되며, 소스코드 생성을 위한 코드 생성기의 입력자료로 사용이 된다.

구조도는 시스템의 프로시듀어들과 이 프로시듀어들의 계층구조 및 이들을 연결하는 자료들로 구성되어, 시스템의 전체적인 모습뿐만 아니라 세부적인 구조까지도 나타나며, 모듈 단위를 효과적으로 정의할 수 있기 때문에 시스템의 일관성을 유지 할 수 있다.

구조도 편집기는 구조도 작성을 위한 편집기능과 작성된 구조도가 규칙에 맞게 작성되었는가를 검증하는 분석기능, 설계된 시스템의 문서화를 위하여 각종 보고서를 출력할 수 있는 보고서 산출기능을 제공한다.

끝으로, 보고서 산출기는 STADE 사용자에게 데이타베이스의 설계명세 내용을 보다 정확하고 간결한 구조적 보고서의 형태로 제공하고, 각 편집기에서 입력한 데이타의 오류 검증도 용이하다. 또한 사용자와의 의사소통도 원활히 하고 다음 단계의 작업자인 프로그래머, 테스트 요원,

유지보수 요원들에게 유용한 문서가 된다.

2.3 코볼정적분석기(COSTAR)

소프트웨어의 품질을 개선하기 위해서는 품질 측정 방법론과 도구가 필요하다. COSTAR는 복잡도와 크기, 사용능률을 중심으로 품질을 측정할 수 있도록 고안되었다. COSTAR는 품질측정의 객관성을 얻기 위해 소프트웨어 개발주기상 거의 끝에서 발생하는 소스코드를 대상으로 삼고 있다.

명세화단계의 산출물들은 작성방법과 표현수단이 다양하여 객관적인 측정기준이 아직 없다. 따라서 사람이 직접 품질측정을 하고 결과를 평가해야 한다. COSTAR는 소스코드중에서도 코볼로 작성된 것만을 대상으로 한다. 이는 이미 개발된 대부분의 프로그램들이 코볼로 작성되어 있고 앞으로도 상당기간 이러한 추세가 지속되리라는 예측에 근거를 두고 있다.

COSTAR 구현에 사용된 기술은 어휘분석, 구문분석, 어의분석 등을 주축으로하는 컴파일러 분야와 ELOC, Cyclomatic Complexity, Software Science 등을 중심으로 하는 복잡도 측정 분야로 요약된다. 이밖에 오류진단을 목적으로 하는 변수사용 분석기법들이 포함되어 있다.

COSTAR의 운용을 중심으로하는 주요기능을 살펴보면

- 사용자접속 : 유닉스 쉘을 그대로 사용하거나 X-Window 환경의 선택메뉴를 이용한다.
- 정적 분석 : 어휘분석, 구문분석, 어의분석, 오류진단 등을 수행하기 위해 변수 및 상수 테이블을 유지하고 프로그램 제어구조를 추출하며 데이타의 정의, 참조, 흐름에 관한 정보를 저장한다.
- 품질 측정 : LOC (원시프로그램의 라인수로 주석라인은 포함되나 복사(Copy)되는 라인은 제외), ELOC (프로시쥬어 디비젼내의 실제 실행될 수 있는 문의 수), Cyclomatic Complexity (프로그램을 제어그라프로 표현한뒤 엣지의 갯수에서 노드의 갯수를 빼고 2를 더해 구한다. 이 값은 프로그램이 순차 구조를 가지면 1이 되나 결정구조가 추가되는 경우 결정구조마다 1씩 추가된다. 하나의 프로그램당 이 수치가 50을 넘으면 복잡하다고 할 수 있다), Halstead의 Software Science(어휘의 종류와 빈도에 의해 난이도와 노력을 계산한다. 어휘는 연산자와 피연산자로 구분하는데 코볼의 경우 문이 연산자, 변수와 상수가 피연산자에 해당된다.) 등을 포함한다.

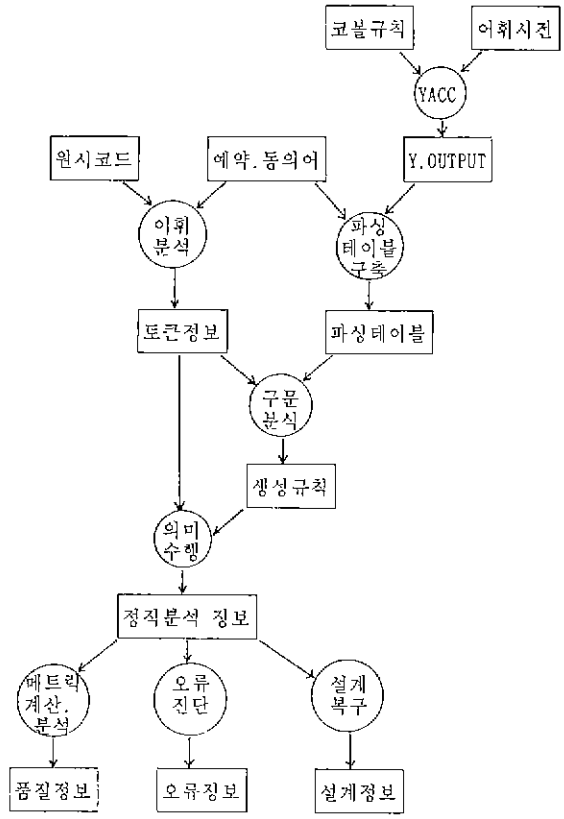


그림 3 정적분석기의 구조[1]

이밖에 사용화일의 갯수, 파라그라프 갯수, 시스템별 메트릭의 집계 등의 추가정보를 생성한다. 정적분석기의 구조는 그림 3과 같다.

2.4 형상관리 지원도구(COMET)

완전한 소프트웨어를 개발하려면 개발주기를 따라 발생하는 모든 정보를 추적하고, 분석하여

저장한뒤 검색할 수 있어야 한다. COMET은 형상식별(Configuration Identification), 변경관리 (Change Control), 버전통제(Version Control)와 형상정보관리(Configuration Status Account-ing) 등의 기능을 통해 이런 목적을 달성시켜 준다. COMET 시스템의 구성도는 그림 4와 같다.

COMET의 형상식별은 형상항목(Software Configuration Item)의 분류를 통하여 이루어진다. 인식되는 각형상항목은 개발주기의 특정단계에서 주로 발생하는 특정유형들로 구별된다. 특정 형상항목은 해당유형에서 발생된 순서대로 일련번호가 부여된다. 이같은 방법으로 소프트웨어 개발주기를 따라 발생하는 모든 정보들은 ID를 부여받게 된다.

형상항목이 식별되어 등록된뒤 일정시간이 흐르면 변경요구가 발생하게 되며 변경요구가 발생하는 원인은 여러가지가 있다. 하드웨어의 변화, 시스템소프트웨어의 변화, 사용자접속 기술의 변화 등과 같은 외부환경의 변화와 사용자요구의 변화, 설계상의 오류 등과 같은 내부환경의 변화로 나누어진다. 이러한 변경정보는 유형별로 정리되어야 개발손실과 오류를 방지할 수 있다. COMET은 변경요구가 발생할 때마다 유형별로 분석하여 저장한 뒤 실제로 실행시에는 이러한 요구를 병합하거나 분리하여 작업지시서를 발행할 수 있게 한다. 작업지시서 (COMET에서는 변경실행서) 에는 변경할 형상항목들이 열거되며 각 해당 버젼들이 지정된다.

변경요구에 의해 다시 등록되는 형상항목은 버젼이 올라간다. 그리고 버젼이 거듭 바뀜에따라 하나의 형상항목에 많은 버젼이 생겨나 관리가 필요해진다. COMET은 버젼들을 줄기(main)와 가지(branch)로 구분하고, 기억장소의 절약을 위해 정원개념과 델타메카니즘을 사용한다. 재구성 알고리즘은 역진법을 기반으로 수행 시간 단축에 역점을 두었다. 개개의 형상항목이 통제되는 것과는 달리 소프트웨어 버젼에 대한 통제는 엮기(thread)에 촛점이 놓여진다. 특정 버젼의 구성에 엮일 형상항목은 무엇이고 해당버젼은 몇번인가 하는 정보가 관리된다. 이러한 버젼은 시스템생성, 품질검사, 방출 등의 작업시 참조된다.

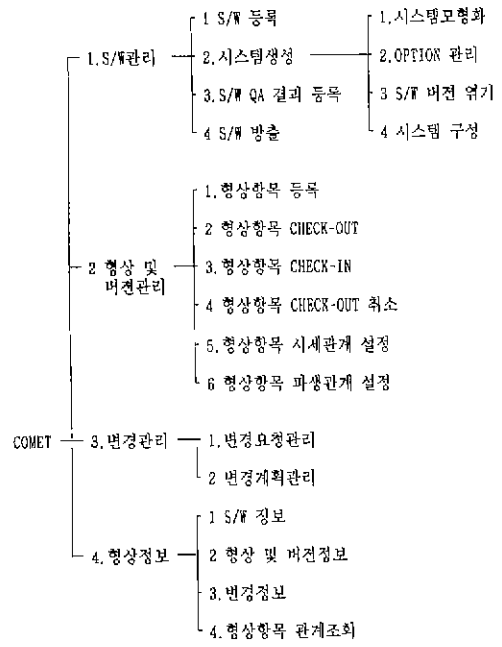


그림 4 COMET 시스템의 구성도[2]

COMET이 이러한 기능을 수행하는데 있어서 가장 필요한 것은 정보의 추적, 분석, 저장, 검색을 도와주는 형상정보관리 기능이다. COMET에서는 Check-In, Check-Out과 각종 등록기능을 통해 정보를 추적하고 분석하며 약간의 사용자접속을 통해 검색기능을 제공한다.

3. 통합 CASE(들무새)의 구성 및 구현 방향

향후 CASE시장[3]은 2가지 형태의 방향으로 나누어 갈 것으로 보이며 그 하나는 전체 생명주기를 지원하는 완전한 Architecture를 목표로 하는 그룹과, 다른 하나로서 분석 및 설계 단계에 촛점을 맞춘 그룹이 있는데 이경우는 현재 보편화 단계에 있기 때문에 앞으로 아주 저렴한 가격대에서 보급될 전망이다.

그런 가운데에도 Taxas Instrument, KnowledgeWare, Intersolve 등을 비롯한 몇몇 선두그룹 공급자들이 전체 생명주기를 지원하는 새로운 분야에 몰두하고 있으나 아직 누구도 모든면에서 완벽하지 않는 상태이며 당분간 전체 생명주기를 완벽하게 지원하는 CASE도구는 출현하기 어려

우며 단기적으로는 Best-of-breed 즉, 나타난 것 중 가능한 최선의 방법을 모으는 접근방법을 써야 할 것이다. Best-of-breed 아키텍춰는 실시간 Meta-data교환을 지원하면서 Versioning 및 도구 자체의 기능성을 충분히 갖춘 정보저장소를 필요로 하나 현재로서는 독립적 정보저장소가 아직 성숙되지 못하는 상태이다.

3.1 시스템 구성

들무새 시스템은 국내에서 처음 시도되는 I-CASE로서, 국산 주전산기 및 워크스테이션에서 COBOL 또는 C 언어를 이용하여 개발되는 응용프로그램을 지원하는 통합 개발환경을 구축함을 목표로 하여 요구분석 단계로부터 소스코드 생성단계까지 지원하는 통합도구로서, 명세화 기법의 자동화와 이들 사이의 변환 기능을 개발하고 분석 및 설계 명세로부터 코드를 자동생성하는 분야와 각 단계별 작업을 통합하기 위한 정보저장소 및 이를 관리하는 사용자 접속기 를 설치하며, 사용자 접속의 편의를 위한 접속부 관리기 및 코드 자동생성 기능등이 있다.

3.2 명세화 도구

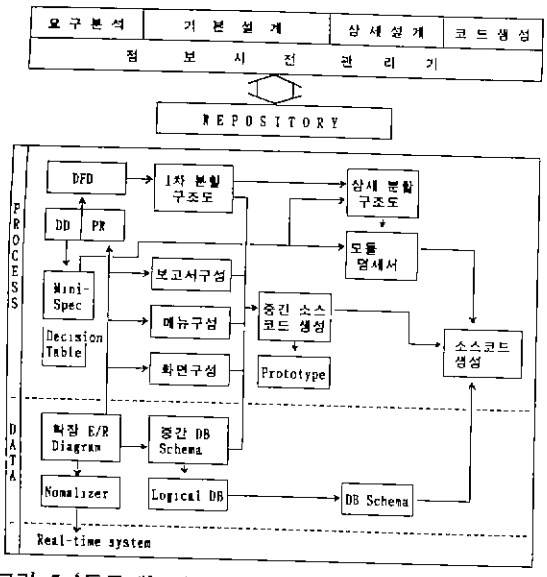


그림 5 '들무새' 시스템 구성도

분석 및 설계단계에서의 요청사항으로서는 우선, 최종 사용자의 사용이 쉽고 실질적 생산성 향상이 이루어져야함으로 그래픽 기능의 도구이여야 하며 메뉴체계에 대한 신중한 연구가 있어야 한다. 다음으로, 재사용 및 유지보수를 감안한 설계가 가능해야 함으로 복수 사용자를 가능케 하는 자료저장소 형태를 갖추어야 한다.

마지막으로 사용 위험성 감소 및 유연성이 제공되어야 하며 전체적인 개발노력과 비용절감이 이루어져야 한다. 이를 위해선, 전 시스템 생명주기를 지원하는 통합된 도구이어야 하며 시스템에의 새로운 기술 수용을 통하여 궁극적으로 최종 결과물인 소스코드를 자동으로 생성할 수 있는 중간정보를 제공하는 역할을 함으로써 한국적 개발 방법론으로 정착될 수 있어야 한다.

상세 설계단계를 지원하는 기법중 Action Diagram이 나름대로 우수하다는 점은 지금까지의 자체조사에서 나온 결과이며 단지, 'C'선언부에 대한 정보를 포함할 수 있게 하는등 언어환경에 적절히 변형된 형태를 취할 것이며 분할구조도의 모듈별 알고리즘을 이곳에서 완성한다. 또한 방법론 사이의 자동변환 기능을 높이기 위해 요구분석 단계에서 작성되는 기능명세 작성 방법을 모듈 Spec. 작성 방법과 동일기법을 쓰되 프로세스의 골격 수준을 작성하게 될것이다.

정보의 표현 기법으로서 널리 알려진 Entity Relationship모델을, 데이터 모델링을 위한 대상 방법론으로 하여 Entity에 대한 Attribute, Data type 등을 정의함으로써 데이터의 관리 및 표준성을 유지하고 각 Entity는 자료사전 및 DFD

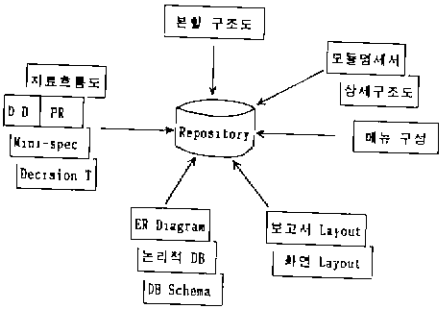


그림 6 분석 및 설계명세화 도구군

작성시 참조됨으로써 Data와 Process간의 연관관계를 파악할 수 있다. 이때 생성된 개체관계도에 대해 정규화를 거쳐 논리적 DB를 생성하고 최종적으로 데이터베이스를 설계하는 과정으로 연결된다.

한편, 최근들어 국내서도 새로이 요구되고 있는 실시간(R/T) 시스템 개발을 지원하기 위한 모델링 기능의 제공을 위하여, 기존 R/T 방법론을 비교 분석한 결과 Visualism과 Logic에서 기반한 Formalism을 결합한 Visual formalism 형태를 갖춘 방법론을 개발하고 이를 Tool Prototype으로 구현할 예정이다.

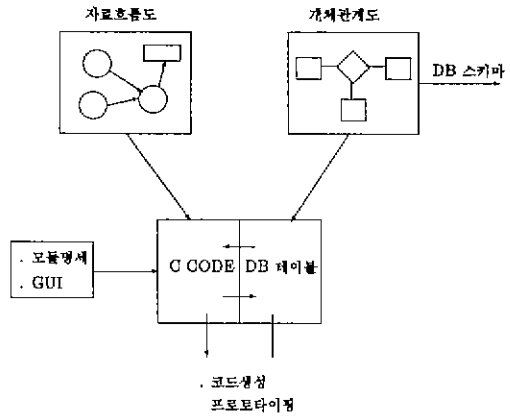


그림 7 자료흐름도와 개체관계도 및 모듈명세로부터의 코드생성시스템

3.3 소스코드 생성 시스템

들무새 시스템에서 생성하려고 하는 코드생성 시스템은 두 단계로 나누어서 행해진다. 첫째는 자료흐름도와 개체관계도를 이용하여 전체 시스템의 호출윤곽 및 대상 시스템의 데이타베이스 스키마를 나타내는 코드부문과 좀더 정형화된 액션다이아그램(action diagram)을 이용하여 만들어진 모듈명세로부터 각 모듈에 대한 코드를 생성해내는 것인데 이를 첫단계에서 만들어진 코드가 호출하여 사용하고 그 밖에 사용자 인터페이스등에 관한 프로그램을 기존에 만들어진 코드에 삽입하여 점진적으로 완성된 시스템으로 발전해 나가는 방법이다.

자료흐름도는 전체 시스템을 세분화하여 프로그래머가 프래그래밍을 할 수 있는 단계까지의 전체 구조를 나타내주는 것으로서 자료사전과 자료저장소, 외부실체 및 외부와의 연결을 나타내주는 외부실체로 구성되어 있다. 자료흐름도는 개념도(context diagram)을 제외한 모든 도면이 그 도면의 모든 기능을 대표하여 나타내는 상위의 프로세스를 가지고 있어서 이러한 체계적인 표현 방법을 나타낼 수 있는 언어를 개발하여 단위별로 생성된 코드를 결합하여 전체 시스템의 코드를 생성해 나가면서 언제든지 추가된 모듈 및 코드에 대하여 쉽게 변경 가능하도록할 계획이다. 이를 위하여 자료흐름도의 기능 프리미티브(functional primitive)의 입출력 자료흐름을 한 모듈의 매개변수로 나타내어 입력 자료흐름을 모듈의 입력 매개변수로 출력자료흐름을 모듈이 수행하고 되돌려 주는 값으로 정의하여 코드를 생성한다.

개체관계도를 이용하여 한 개체(entity) 혹은 관계(relationship)가 가지고 있는 속성에 대하여 이를 데이타베이스에서 테이블의 속성으로 대응시켜 대상 시스템에 대한 스키마를 생성하고 각 테이블에 대한 실제 데이타를 입력하여 이를 자료흐름도의 기능프리미티브에 대응시켜 자료흐름도로부터 생성된 실행용 코드를 이용하여 대상 시스템의 구조 및 데이타베이스 및 모듈명세가 잘되어 있는지 신속하게 검증할 수 있는 프로토타이핑 시스템도 아울러 개발해 나갈 계획이다. 그림 7은 자료흐름도, 개체관계도 및 모듈명세로부터 코드를 자동적으로 생성하기 위한 개괄적인 윤곽을 나타낸다.

3.4 GUI(모자이크) 도구

3.4.1 모자이크 시스템의 기본모형

지금까지의 GUI관련도구들은 일반적으로 두가지 유형으로 분류될 수 있으며 대화기법들을 모은 라이브러리인 사용자접속 툴킷과 사용자접속 개발지원 시스템이 그것이다[5].

모자이크는 Motif 표준을 따르는 사용자접속의 개발을 지원하는 GUI 도구로서 UNIX, X Window,Motif 환경에서 운영되며, 최종적으로는

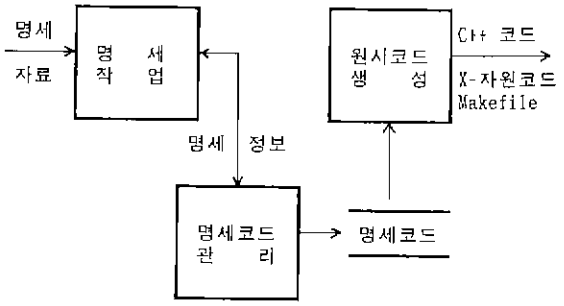


그림 8 모자이크의 자료흐름도[6]

Motif 라이브러리를 이용하는 컴파일 가능한 C++ 원시코드 군을 생성한다. 일반적으로, 그래픽 사용자접속의 개발을 지원하는 도구류는 필수적으로 다음 세가지 기능을 제공해야 한다. 첫째, 사용자가 대화식으로 시각적 환경에서 작업 할 수 있는 명세 작업대를 들 수 있다. 둘째, 사용자의 작업 내용을 표현하여 가역적으로 관리할 수 있는 고수준의 명세언어를 들 수 있으며, 마지막으로 명세언어로 표현된 코드를 최종의 컴파일 가능한 언어로 전환하는 코드 생성기가 필요하다.

모자이크는 이와같은 필요성을 만족하는 세부분으로 구성되어 있으며, 이들 세개의 하위 시스템들은 명세언어 코드를 중심으로 그림 8과 같이 협동하여 최종의 컴파일 가능한 원시코드 군을 생성하며, 고수준의 명세언어로 기술된 명세코드를 이용하여 사용자의 작업을 가역적으로 지원한다.

3.4.2 사용자의 명세작업 모형

사용자의 목적 소프트웨어에 대한 명세작업을 지원하기 위한 모형이다. 사용자의 명세작업은 개발하고자 하는 소프트웨어를 정의하는 작업으로, 사용자접속과 관련된 명세, 응용과 관련된 명세, 그리고 자료(Data)와 관련된 명세를 작성하는 작업의 세가지 유형으로 분류할 수 있다. 그러나 모자이크에서는 우선 사용자 접속의 명세를 중심으로, 사용자접속 명세와 응용 명세의 두가지로 사용자의 작업을 모형화 하였다. 사용자접속 명세는 그래픽 편집기에서 지원하고, 응용영역 명세는 코드 편집기에서 지원하도록 구성되었다. 이러한 명세 작업의 결과로서 명세 코드를 생성할 수 있는 명세정보가 구축된다.

3.4.3 사용자 명세정보의 관리 모형

사용자 명세의 표현 모형이란 사용자가 제공한 명세정보를 작업의 가역성 및 표현의 충분성을 기반으로 하여 고수준의 명세 언어로 표현하여 관리하기 위한 모형이다. Motif 모형을 중심으로 사용자 접속의 개발 작업을 표현하기 위한 명세언어는 두가지 모형을 표현할 수 있어야 한다. 하나는 핵심이되는 Motif 모형이고, 다른 하나는 작업 정보의 연관관계를 표현하는 프로젝트 정보 모형이다. 후자의 모형은 작업 가역성을 확보하기 위하여 필요한 부수적인 모형이다. 전자는 다시 접속 객체 및 그들 사이의 관계를 표현하는 부분과, 계면과는 독립적인 응용코드 부분으로 나눌 수 있다. 따라서 명세 정보는 세가지로 구분할 수 있으며 우리는 이들을 프로젝트 정보, 응용 정보, 접속 정보라고 정의 하였다. 모자이크에서는 이러한 세 부류의 정보를 객체지향적 표현방식에 기반을 둔 X-자원코드의 형식으로 표현한다.

3.4.4 원시코드의 생성 모형

원시 코드군의 생성 모형이란 명세정보 또는 명세 언어 코드로부터 컴파일 가능한 원시코드 군을 생성하기 위한 모형으로, 그 결과물은 C++ 코드 군이다. 사용자 접속의 모양에 대한 요구사항 만을 추출하고자 하는 프로토타이핑 도구라면 원시 코드를 반드시 생성할 필요는 없을 것이다. 그러나 코딩이라는 지루하고 반복적이며 오류를 유발하기 쉬운 작업을 자동화 할 수 있다면, 코드 생성은 필수적인 기능 요소이다.

생성을 통하여 얻은 코드는 일반적으로 디버깅이 어렵고, 수행 효율성이 떨어지는 단점이 있으나, 반면에 개발기간의 단축(40-50%)[7] 코드의 일관성 유지, 고수준 언어에 의한 편리한 유지보수 등의 상당한 장점이 있다. 모자이크에서는 명세언어 코드로 부터 접속과 관련된 C++ 코드군과 X의 자원화일, 응용과 관련된 C++ 코드군, 그리고 이들을 엮어서 컴파일이 가능토록 하는 Makefile을 생성한다. 따라서 사용자는 최

종적으로는 Makefile 만을 수행시키면 된다.

3.5 정보저장소

개발정보의 독립성을 확보하기 위해서는 개발주기의 모든 단계에서 발생하는 정보들에 대한 모형화가 필요하다. 개발정보의 모형은 개발될 소프트웨어가 다룰 현실세계의 정보들에 대한 모형들을 다시 모형화하므로 메타모형이 된다. 들무새에서는 Peter Chen의 개체-관계 모형화 기법을 이용하여 개발주기상의 모든 정보들의 개체관계도를 작성하였다. 개발정보를 위한 메타모형이 다른 정보모형과 다른 점은 각 개체들 이 시간의 흐름에 따라 관계를 맺는 점이다. 예를 들면 자료흐름도가 생성되어야 모듈간의 관계도를 생성할 수 있고 미니명세서가 만들어져야 모듈명세서가 생성될 수 있는 등의 시제관계이다. 엄격하게 말하면 개발초기의 모든 정보들은 하나의 소프트웨어에 의존적이다. 그리고 시간이 흐르면서 추가되는 정보들은 선행정보들에 의존적이다. 그러나 변경이 발생하면 이러한 관계가 잘 유지되지 않는다. 그것은 변경이 개발주기의 뒷쪽이나 유지보수단계에서 주로 발생하기 때문이다. 들무새에서는 이러한 점을 고려하여 개발주기상에서 발생하는 각정보를 가능한한 독립적으로 정의하였다. 정보저장소의 구조는 그림 9와 같다.

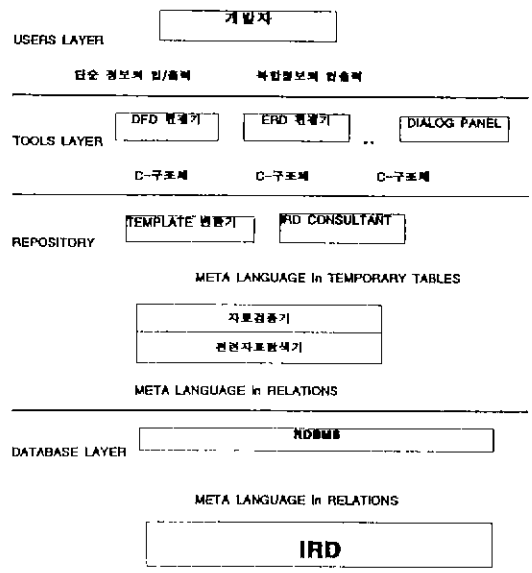


그림 9 정보저장소 구조

개발정보의 독립성과 더불어 중요한 것은 정보의 행태를 표현하는 프로세스의 독립성이다. 들무새에서는 이러한 프로세스의 독립성을 확보하기 위해 모든 개발정보들을 나열하고 각 개발정보가 취할 수 있는 모든 프로세스를 정의하여 정보저장소의 써비스라이브러리에 등록시킨다.

정보저장소가 탄력성을 가지려면 시스템의 다른 계층과의 접속을 데이타로만 하여야 한다.들무새에서는 단위도구와 정보저장소간의 접속을 C-구조체로 하도록 설계하였다. C-구조체는 단위도구가 사용자로부터 입력받은 정보와 처리메시지를 포함한다. 단위도구로 부터 넘겨받은C-구조체는 Template 변환기에 의해 메타언어로 변환된다. 메타언어는 정보저장소의 하부구조에 저장될 메타데이타의 생성규칙을 정의한 문법에 의해 검증된다. 들무새에서는 자료검증기가 이러한 검증기능을 수행한다. 발생순서에 따라 이루어지는 자료변환규칙과 탐색알고리즘은 관련자료탐색기가 수행한다. 관련자료탐색기는 여기에다 관계형 데이타베이스 시스템과 접속하는 기능도 수행한다. 접속방법은 SQL을 사용하며 데이타는 메타언어의 형태로 넘겨진다. 따라서 데이타베이스 시스템으로 부터 독립적으로 설계하는 것이 가능하다.

저장된 개발정보는 IRD Consultant를 통하여 검색되며 단위도구와의 접속은 역시 C-구조체를 이용한다. 한편 정보저장소 자체의 Dialog Panel을 통하여 사용자가 직접 정보저장소를 뒤져볼 수도 있고 전체적인 변경이나 삭제의 작업도 가능하다.

4. 결 론

본고에서는 시스템공학연구소에서 개발한 4개의 도구 즉, 자료흐름도의 편집 및 분석기능을 가진 START, 구조적 설계를 위한 STADE, COBOL코드의 정적 분석을 위한 COSTAR 및 버젼 및 변경관리를 위한 COMET에 대하여 그 기능 및 특성을 간략히 설명하였다. 우리 연구소에서

는 이 4개의 도구를 개발한 경험을 바탕으로 하여 소프트웨어의 생명주기 전체를 지원할 수 있는 도구 즉, 들무새 시스템을 개발중에 있는데 그 주요 도구를 살펴보면 사용자의 요구명세를 표현하기 위하여 사용되는 모든 모델링 도구 및 그래픽 지원도구를 정형화하여 사용자에게 요구분석의 편리함과 정형성을 제공하여 주기 위한 명세화 지원도구와, 여기에서 제공된 요구명세를 유기적으로 연결하여 전체 시스템의 윤곽 및 좀 더 정형화된 모듈명세로부터 코드를 이용하여 점진적으로 코드를 생성해주는 소스코드 생성도구, Motif를 이용하여 사용자 인터페이스를 지원하는 GUI도구 및 전체 시스템에 대한 정보를 관리 총괄하는 정보저장소 등으로 나누어져 있다. 들무새 시스템은 약 3년여에 걸쳐 개발될 예정이며 통합 CASE도구로서의 면모를 갖추어 나갈 수 있도록 개발할 계획이다.

참고문헌

[1] 과학기술처, "소프트웨어 테스팅 도구 개발," pp. 221, 1991. 7.

[2] 과학기술처, "소프트웨어 형상 및 변경관리 시스템에 관한 연구(III)," pp. 278, 1993. 10.

[3] GartnerGroup, "A five-year plan to overcome Applications development and management challanges," pp. 15~1~10 Jul. 1993.

[4] GartnerGroup, "A five-year plan to overcome Applications development and management challanges," pp. 15~5 Jul. 1993.

[5] B. A. Myers, "Userinterface tools: Introduction and survay," IEEE Software, pp. 15~23, Jan. 1989.

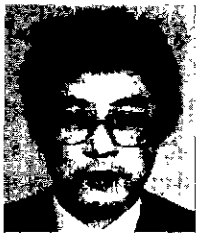
[6] 노영주 외 "Motif 표준을 지원하는 사용자 계면 합성기 개발: 모자이크 모형," pp. 826, 1993. 10.

[7] K. J. Shmucker, "MacApp: An Application Framework," BYTE, pp. 189~193, Aug. 1986.

이 단 형

1971 서울대학교 공과대학 학사
1983 Authur D. Little 경영과학 석사
1990 Verginia Commonwealth Univ. 정보시스템 박사
1972 ~현재 시스템공학연구소 책임연구원
1992 ~현재 고려대학교 전산학과 객원 교수
관심 분야 : Software Eng., Decision Science, Artificial Intelligence

신 규 상



1981 성균관대학교 통계학과 학사
1983 서울대학교 계산통계학과 석사
1983 ~현재 시스템공학연구소 소프트웨어공학연구부 선임연구원
관심 분야 : Software Engineering, 4GL, DBMS

이 승 지

1983 전남대학교 공장설계학과 학사
1992 아주대학교 전자계산학과 석사
1983 ~현재 시스템공학연구소 소프트웨어공학연구부 선임연구원
1991 ~1992 IBM Sterling forest 객원교수
관심 분야 : Software Engineering, Project Management