□기술해설□

IRDS에 기반한 CASE 도구 정보저장소 (Repository)의 구현 사례

충남대학교 진성일*
해전전문대학 김기봉**
시스템공학 연구소 박중기** · 이혜란**
충남대학교 조유희***

목 차

1. 서 론

최근 컴퓨터를 이용한 정보처리가 급속히 발전함에 따라 조직체에서 사용하는 방대한 양의 데이타를 효율적으로 저장하고 관리하기 위한 데이타베이스 관리 시스템(DBMS)의 사용이 크게 늘고 있다.

더우기 근래에는 의사결정을 도와주는 전문가 시스템, 고품질의 문서를 작성하는 탁상출판 시스템, 소프트웨어 생산성을 높이는 Computer-Aided Software Engineering(CASE) 도구 등의 각 응용 시스템과 DBMS가 연계되어 서로의 정보자원을 공유하며 통합된 정보시스템을 구축하는 경향이 있다.

그러나, DBMS의 기능은 조직체내의 인사정보, 재고정보, 회계정보 등의 단순한 데이타 관리의 수준에 그치고 있으며, 전 조직체의 정보관리를 위해 필요한 경영정책, 조직구조, 업무규칙, 하드웨어 및 소프트웨어 환경, 생명주기 관리 등의 메타데이타인 정보자원의 관리에는 그 기능이 미치지 못하고 있다.

특히 현재 개발된 대부분의 CASE 도구들은 소프트웨어 생명주기 각 단계의 일부분만을 지원하고 있기 때문에 소프트웨어 개발의 완전 자동화를 이룰 수 없다. 이러한 문제를 해결하기 위해 최근에는 통합 CASE 도구에 대한 요구가 높아지고 있으며, 이러한 통합 CASE 도구를 구성함에 있어서 데이타의 공유는 가장 핵심적인 문제중의 하나이다.

따라서 생명주기 각 단계를 지원하는 도구들에 생성되는 정보를 일관성 있게 저장하고 공유하는 역할을 효과적으로 지원하기 위해서는 정보저장소를 필요로 한다.

CASE 도구의 정보저장소는 소프트웨어 전체 생명주기 동안 모아진 시스템 정보가 관리

*종신회원
**정 회 원
***학생회원

되는 곳으로서, 각 도구들, 각 개발 단계들, 사용자들, 응용 프로그램들 사이의 시스템 정보를 공유할 수 있도록 해준다. 즉, 단위 CASE 도구들의 한계를 벗어나 각 도구들이 유기적인 관계를 맺어서 전체 도구들이 가지고 있는 기능을 극대화시키기 위한 CASE 도구 통합 방법의 하나라고 할 수 있다.

이러한 정보자원을 효율적으로 정의하여 통합, 저장, 관리 및 공유할 수 있는 기능을 제공할 수 있는 시스템이 바로 Information Resource Dictionary System(IRDS)이다. IRDS는 정보 자원과 메타 정보 관리를 위한 핵심이 되는 소프트웨어 도구로서 정보 환경을 표현하거나 모델링할 수 있도록 하는 특별한 응용시스템이다.

IRDS는 CASE 도구를 위한 메타 정보를 Information Resource Dictionary(IRD)라고 불리우는 공유가능한 정보저장소에 저장하고 IRD를 이용하여 IRDS를 접근하고 관리한다. 이러한 IRDS를 이용함으로써 서로다른 시스템 간의 메타데이타의 교환(transportability)이 가능하며, 스키마 확장성 및 전체 생명 주기 지원 기능을 통해 사전 시스템의 질을 향상시킬 수 있다.

본 고에서는 CASE 도구 정보저장소의 개념 및 현재 표준화 동향에 대해 살펴보고, 표준화의 하나인 IRDS의 구조, 기능 및 응용분야에 대해 살펴본다. 또한 구현 사례로써 현재 ANSI (American National Standard Institute) IRDS 표준안에 근거하여 개발한 IRDS 서비스/사용자 인터페이스에 대해 살펴보고, CASE 도구에서 IRDS 명령어를 라이브러리 함수들의 인수로 전달하여 C 언어 상에서 사용할 수 있는 CLI(Call Level Interface) 형식의 인터페이스에 대해 살펴보고, 소프트웨어 개발 과정 중 시스템 분석 및 설계 단계에서 생성되는 산출물을 분석하여 정보저장소에 저장하기 위한 IRDS 모델링에 대해 기술하고 실제 예를 통해 생명주기 단계간의 상호 참조 추적이 가능함을 보였다. 마지막으로 결론 및 향후 연구 과제에 대해 언급한다.

2. CASE 도구 정보저장소

정보저장소는 소프트웨어 개발의 각 단계에서 사용되고 생산되어지는 모든 정보와 시스템 요소들을 저장함으로써 시스템 분석 단계에서부터 유지 보수에 이르기까지 전체 소프트웨어 생명주기를 지원한다. 또한, 정보저장소는 이러한 많은 형태의 정보뿐만 아니라, 정보들의 상호관계와 이들 정보들의 유효성을 증명하고 사용하는 규칙도 포함하여 시스템의 이해와 변화 관리를 지원한다[7, 15, 18].

2.1 정보저장소 개념

정보저장소는 소프트웨어 생명주기 동안 모아진 모든 시스템의 정보가 관리되고 저장되는 곳으로서, 각 도구들, 각 개발 단계들, 사용자들, 응용 프로그램들 사이의 시스템 정보를 공유할 수 있도록 해준다.

정보저장소에 저장되는 정보의 유형으로는 논리적 자료 및 처리 모형, 물리적 정의와 구현 코드, 기업 모형, 프로젝트 자료 규칙, 개체간의 관계, 정보저장소 메타모델과 검증 규칙, 프로젝트 관리와 감사 정보 등이 있다[13, 19].

통합 CASE 도구의 정보저장소 구현에 있어서 통일된 정보의 작성규칙과 정보처리 규정이 필수적이다. 각 단계별로 발생하는 개발정보를 정해진 규칙과 처리규정에 의해 분석, 가공, 저장 및 검색할 수 있도록 지원하는 하부 구조가 필요하다. 이러한 하부 구조는 여러가지 형태가 있는데 그 중 가장 효율적인 것이 CASE 도구의 전용으로 설계되는 DBMS로써 CASE 정보저장소를 구축하는 것이다.

이러한 정보저장소가 시스템 정보를 안전하게 공유하고 자동화된 방법으로 관리하기 위해서는 통일된 작성규칙과 처리 규정이 유지되어야 하며 이를 위해 요구되는 기능으로는 정보의 접근에 대한 보안성 관리 기능, 정보의 생성, 변경, 삭제 등의 기록을 유지하고 과거의 정보를 재생할 수 있도록 지원하는 버젼관리 기능, 정보의 변경에 따른 효과를 분석하고 변경과정을 기록하는 변경관리 기능, 정보모델의 어의적인 분석을 하고 정보와 정보간의 유기적인 관계를 알기 쉽게 표현하여 축적된 개발정보의 재활용성을 높여주는 형상관리 기능, 저장된 정보를 사용자의 관점에 맞게 검색할 수 있도록

지원하는 검색 기능, 여러 사용자가 동시에 저장된 정보를 이용할 수 있도록 제어하는 기능, 각 도구들이 정보저장소의 정보 관리 기능을 사용할 수 있도록 인터페이스에 관한 정보를 제공하며, 하나의 정보저장소로 부터 다른 정보저장소나 데이타베이스 또는 자료사전으로의 정보 교환도 가능케 하는 개방형 기능, 저장된 내용의 정확성을 검증하는 검증 규칙을 정보저장소 메타모델에 포함시킴으로써 저장정보가 정보저장소에 등록되거나 변경될 때 지켜야하는 제한조건을 각 도구들에서 통일된 방법으로 단순화시킬 수 있는 기능 등이 필요하다[15]. 따라서 이러한 요건들을 만족시키기 위해 현재 관계형 데이타베이스를 이용한 정보저장소의 구성, 자료사전의 구현, 정보저장소의 표준화, 지능형 정보저장소 등에 관한 연구가 활발하다.

2.2 표준화 현황

다양한 구조와 기능을 가진 정보저장소들을 사용함으로써 이들 정보저장소를 이용하는 응용시스템들은 정보를 공유하기도 어려웠을 뿐만아니라 통합된 응용시스템을 구성할 수 없다. 또한 미래 정보자원 환경이 더 많은 종류의 정보를 각각의 정보처리 시스템간에 공유할 수 있도록 변화함에 따라 통일된 규격을 갖는 정보저장소가 필요하다[11].

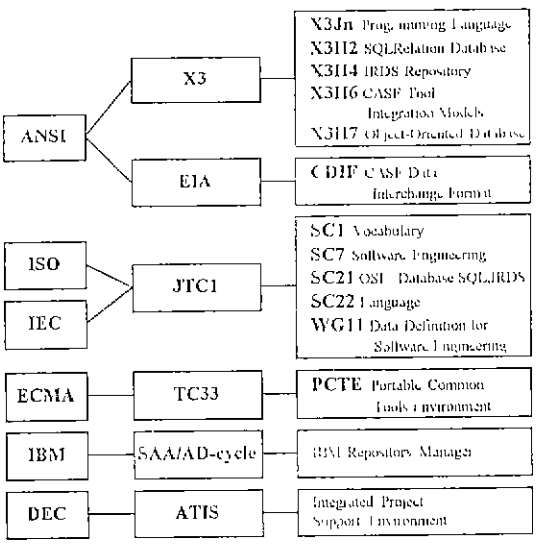


그림 1 정보저장소 및 CASE 표준화 기구 및 위원회

이에 따라 국제적인 표준화 단체들은 정보저장소 표준화에 노력을 기울이고 있으며, 이러한 표준화 단체들은 그림 1과 같이 미국 국립 표준국(ANSI), 국제 표준화 기구(ISO), 국제 전기 표준회의(IEC), 유럽 전자계산기 공업회(ECMA)를 중심으로 이루어 지고 있다. 또한, 업체로는 IBM과 DEC에서 자체 정보저장소를 상품화하여 시스템 개발자에게 제공하고 있다.

2.2.1 CASE Data Interchange Format (CDIF)

CDIF는 CASE 도구들 간의 정보 공유를 가능하게 하기 위한 언어에 대한 명세이다. CDIF는 다른 곳에 있는 정보를 정보저장소로 가져오거나, 정보저장소의 정보를 CASE 도구에 보낼때 유용하다. CDIF가 도구들 간의 정보 공유를 위해 사용하는 수단은 CASE 도구에서 출력되거나 CASE 도구에 입력되는 ASCII 문자화일이다. 이 화일 자체는 정보에 대한 설명뿐만 아니라 시스템 정보도 포함하고 있으며, 문자뿐만 아니라 그래픽 정보도 가지고 있다[11].

2.2.2 Portable Common Tool Environment (PCTE)

PCTE는 소형 컴퓨터의 LAN 환경에서 CASE 도구의 이식성을 높이기 위한 목적을 가진 인터페이스 표준이다. 이는 운영 체제와 CASE 도구 사이에 수행되는 함수들의 집합을 명시함으로써 CASE 도구가 운영 체제를 호출하기 보다는 이 함수들을 호출하도록 하는 것이다.

PCTE가 갖는 특징으로는 문자와 고해상도의 그래픽을 모두 포함하는 다중 윈도우 관리 기능, 서로 연결되어 있는 어떤 워크스테이션에서라도 명령의 수행이 가능하도록 하는 분산처리 기능, 분산 저장과 연결된 워크스테이션으로 부터의 정보 추출 기능 등이 있다[11].

2.2.3 IBM Repository Manager/MVS

Repository Manger는 IBM이 개발한 응용시스템 개발환경인 AD/Cycle의 모든 생명주기들의 각 단계를 지원하는 소프트웨어 도구들과 서비스를 제공하는 핵심부분이다. 서비스들

의 확장집합은 정보저장소와 인터페이스할 수 있도록 제공해 준다. 정보저장소내에 저장된 정보는 임의의 도구에 의해 접근되어지고 Repository Manager에 의해 표현되는 형식에 의해 조작된다. 결국 이러한 정보는 서로 다른 도구들 간에 정보 공유가 가능해 진다.

SAA(System Applicaton Architecture)의 한 요소인 Repository Manager는 SAA 환경 전체에 걸쳐 포괄적이고 통합된 응용시스템 개발의 틀을 제공하는 기반이 된다. 이러한 SAA 환경은 생명주기 전체를 지원, 공통된 사용자 인터페이스, 공통된 정보저장소, 개방형 구조를 제공한다.

Repository Manager는 상업적으로 구입이 가능하며, 도구와 사용자의 새로운 정보에 대한 요구를 수용할 수 있도록 확장이 가능하다[10].

2.2.4 A Tools Integration Standard(ATIS)

ATIS는 Integrated Project Support Environment(IPSE)를 위한 객체 지향적 정보저장소 인터페이스이다. Digital Equipment Corporation에서는 ATIS를 그들의 정보저장소인 CDD/Repository에 합병시켰고, ATIS를 IRDS 정보저장소 표준에 합병시켜야 한다고 제안하였다.

ATIS 표준은 정보저장소의 인터페이스를 정의하고 있는데, 이 인터페이스를 사용하는 정보저장소들은 상호 통신이 가능하다.

ATIS 표준은 SAA 처럼 전체적인 통합의 틀을 제공할 뿐만아니라, 버젼관리, 형상관리, 도구들 간의 통신, 소프트웨어 객체들 간의 관계, 도구의 추가를 위한 동적인 확장성 등을 제공한다 [11].

2.2.5 Information Resource Dictionary System(IRDS)

IRDS는 National Institute of Standards and Technology(NIST)에서 개발한 정보저장소 표준이다. 이것은 정보저장소의 내용, 사용자 인터테이스와 정보저장소 간에 정보를 옮기기 위한 인터페이스 등을 정의하고 있다.

IRDS는 모든 정보저장소의 내용과 입력, 출력을 표현하고 있지만, 기술적인 설계나 구현에 대한 계획을 하고 있지 않아 이는 실제로 정보저장소를 개발하는 제작자들의 책임이다. 각각의 독립적으로 개발된 정보저장소가 기본적으로 같은 정보를 포함하고, 사용자에게 똑같이 보이며, 정보를 공유할 수 있는 것을 보장하는 것이 IRDS 표준의 목적이다. 이러한 것이 모두 만족되면 그 정보저장소는 IRDS 표준을 따른다고 간주한다.

한편, ANSI IRDS가 ISO에 제시되어 ISO JTC/SC21/WG3에 ISO의 IRDS 표준화 작업을 할당하여 표준화가 진행되고 있다.

ISO의 IRDS 표준은 SQL(Structure Query Language)에 바탕을 둔 정보모델링을 채택하고 있으나 ANSI IRDS 표준은 E-R 모델을 택하고 있다.

IRDS 표준은 다음의 여섯가지 모듈을 포함한다. [6, 11, 19]

- Core IRDS(Module 1)

IRDS 명령어를 지정하며, IRD와 IRD 스키마의 내용을 선택하고 유지, 보고하기 위한 패널인터페이스를 제공하고, 버젼제어, Minimal 스키마, Import/Export 기능, IRD 뷰 등 과 같은 기능을 제공

- Basic Functional Schema(Module 2)

많은 IRD 응용 분야에 적용할 수 있는 가장 공통적인 스키마 구조를 기술

- IRDS Security(Module 3)

IRD 및 IRD 스키마에 대한 접근을 제어하기 위한 접근제어 기능과 관련된 데이타 모델 등을 정의

- Extensible Life Cycle Phase Facility (Module 4)

Core IRDS가 제공하는 생명주기 클래스(Uncontrolled, Controlled, Archived) 이외에 계층적 단계 모델링, 관계성 구조(Relationship-sensitivity Structure), 생명주기 무결성 규칙 등을 제공

- IRDS Procedure Facility(Module 5)

IRDS 명령어로 구성된 프로시듀어를 정의하고 실행하기 위한 메카니즘을 제공

- Application Program Interface(Module 6)

응용프로그램의 IRD 및 IRD 스키마를 접근할 수 있는 수단을 제공

3. IRDS

3.1 IRDS 구조

IRDS 구조는 두개의 정보저장소인 IRD와 IRD 스키마로 구성되며, IRD 내의 내용은 실제적으로 관리할 데이타와 어떤 정보가 저장되고, 정보의 정의는 무엇이고, 정보자원의 구조, 사용처, 다른 정보들과의 관계 등을 포함하며, 이들을 생성, 접근, 조작 가능하도록 하는 기능을 제공한다.

IRDS의 구조는 그림 3과 같이 4개의 단계와 인접 단계를 두개씩 묶어 3개의 쌍으로 나타낸다. 이러한 구조는 E-R 모델에 기초를 두고 있으며, 각각의 단계는 타입과 인스턴스 관계로 구성된다. 이러한 타입/인스턴스 개념은 오늘날 응용프로그램이나 DBMS에서 특정 자료의 접근시 자료의 타입이 무엇인지를 결정하고 자료처리를 실행하는 것과 동일한 개념으로 설계된 것이다[1, 2, 4, 9].

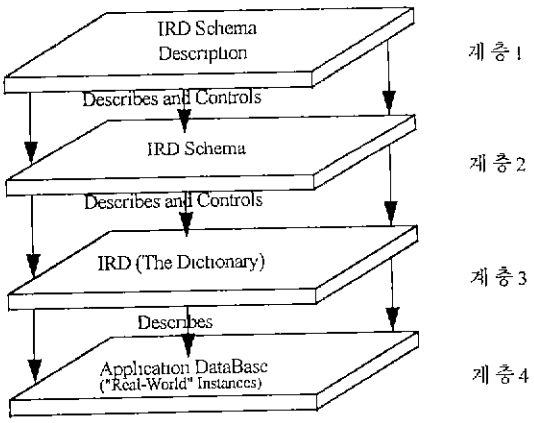


그림 2 IRDS 구조

－IRD 스키마 정의 단계

IRD 스키마 정의 단계에 저장될 정보의 타입을 기술하는 단계로서, IRD에 있는 항목들에 저장되는 항목들에 대한 존재론적 구조를 설명하기 위한 것이며 IRD에 저장할 정보는 반드시 이단계로부터 기술해야 한다.

－IRD 스키마 단계

IRD 스키마 정의 단계에 저장된 타입의 인스턴스를 저장하는 동시에 IRD에 저장될 타입에 대한 인스턴스를 저장하는 단계로서 하나 이상의 IRD를 정의할 수 있으며, 하나의 IRD 정의는 하나 이상의 IRD 스키마를 포함할 수 있다. 이 단계에 저장되는 데이타들은 IRD 구현자나 데이타 모델 설계자, 데이타 관리자들에 의해 저장되고 조작된다.

－IRD 단계

IRD 스키마 단계에 저장되는 타입에 대한 인스턴스를 저장하는 동시에 응용단계에 저장되는 정보에 대한 타입을 저장하는 단계로서 다수의 IRD들은 반드시 하나의 스키마에 의해 기술되어야 하며, 이 단계에 저장될 데이타 타입은 IRD 정의 단계에서 기술하는 것이 가능하다면 정의하는 타입에 대한 제한은 없다. 충분한 IRD가 존재하게 되면 소프트웨어 도구들에 의해 자동적으로 IRD에 데이타를 조작할 수 있게 됨에 따라 다양한 정보를 지원할 수 있다.

－응용 단계

IRD 단계에서 정의한 데이타의 인스턴스가 저장되면 주로 정보시스템의 사용자가 다루는 정보가 된다. 따라서 데이타베이스 관리자나 응용프로그램 작성자들에게 필요한 정보가 되는데, 이 응용 단계에 있는 데이타를 접근하기 전에 우선 IRD 단계에 저장된 데이타 타입을 참조하게 된다.

IRDS가 직접 관리하는 부분은 IRD 스키마 정의 단계, IRD 스키마 단계, IRD 단계이며, 응용단계에 저장되는 데이타는 일반 DBMS나 응용시스템에 의해 저장, 조작된다. 따라서 IRDS는 관리하려고 하는 정보자원의 명세만을 관리하는 소프트웨어라고 할 수 있다.

3.2 IRDS 기능

응용 데이타베이스에 저장되는 정보는 조직에서 평상시 발생하는 정보를 저장하는 반면 IRD에 저장되는 정보는 응용 데이타베이스에서 저장되는 정보의 한단계 높은 정보가 된다. 즉, 데이타 항목에 대한 타입이나 정보처리에 관련된 소프트웨어, 하드웨어 등에 관련된 기술정보가 된다[16].

IRDS는 IRD에 기록된 데이타를 요청하는 사용자나 프로세서에게 질의어나 보고서 같은

적절한 접근 수단을 제공하며, IRD의 자료에 대한 삽입, 삭제, 수정의 기능과 자료의 무결성도 제공한다. IRDS에 관련된 자료는 DBMS의 도움을 받아 저장, 관리될 수 있고, 데이타베이스의 접근을 위하여 IRDS 자료를 필요로 하게 된다.

IRDS는 DBMS가 갖는 기능과 IRDS만이 갖는 고유 기능을 가지고 있다[16].

(1) 일반 DBMS 기능

－제약 조건의 강화(Enforcement of Constraints)

개체에 관련된 값과 개체들 간의 관계에 관한 제약성을 기술할 수 있는 수단을 제공

－접근 제어(Access Control)

접근이 허용된 사용자에게만 IRD내의 자료에 대한 접근을 할 수 있는 수단을 제공

－감사 추적(Audit Trail)

사전, 디렉토리, 모델 관리자 등과 같은 IRD 변화에 대한 감사를 선택적으로 할 수 있는 기능을 제공

－한계값과 묵시값(Limit & Default)

개체들이 생성되거나 변경될 때 특정 속성값들에 대한 한계값, 널값, 묵시값을 기술

－무결성 관리(Integrity Management)

단일 혹은 다수 사용자 환경에서 IRD 또는 IRD 스키마 단계에서 자료의 무결성을 유지

－참조 무결성(Reference Integrity)

개체들 간의 참조 무결성 기능을 제공

－질의 및 보고 기능(Query & Reporting Facility)

IRD내에 저장된 자료에 대한 일반 질의 보고서 기능과 시스템 관리자가 서브시스템의 상태나 성능을 감시

－원격 접근(Remote Access)

IRD간의 클라이언트/서버 연산을 제공

－병행 수행 제어(Concurrency Control)

Ckeck-in이나 Check-out과 같은 IRDS 제어하에 IRD간의 복사를 가능하게 하는 기능

(2) IRDS 고유 기능

－이름 부여(Naming)

IRD에 저장된 특정 개체에 대해 고유의 이름을 부여하여 다른 개체들과 구분을 가능하게 하는 기능

－사전내용의 상태 구별(Status of Dictionary Content)

IRD에 저장된 자료의 형태에 따라 Uncontrolled, Archived으로 구분하여 자료의 상태를 식별할 수 있도록 하는 기능

－생명주기 관리(Life Cycle Management)

정보시스템의 소재와 개발, 사용을 제어하기 위한 다수개의 단계로 구분되는 생명주기 개념을 제공

－버젼제어(Version Control)

개체의 버젼이 생성될 때마다 이를 유지하고 제어할 수 있는 기능을 제공

－분할성(Partitioning)

IRD와 IRD 스키마에 저장되어 있는 정보를 여러개로 분할하여 각각 처리할 수 있도록 하는 기능을 제공

－복사 생성(Copy Creation)

IRD에 저장된 특정 개체를 복사하여 새로운 개체를 생성하는 기능을 제공

3.3 IRDS 응용 분야

IRDS는 정보자원 관리에 필수적인 메타데이타를 관리함으로써 다음과 같은 응용분야에 이용될 수 있다.

－감사

IRDS는 시스템 개발 생명주기 단계에서 필요한 감사와 통제 시스템, 방법론, 테스팅 명세, 문서화 등을 정의하여 시스템 설계의 합법성, 비용/효과 분석, 문서화 표준여부 등에 이용

－컴퓨터 응용

IRDS와 이와 관련된 장비들의 제반 사항들을 정의하여 스케쥴링, 보안절차, 백업과 재저장등의 작업에 이용

－형상관리

IRDS는 형상기록 정보를 기술하여 시스템의 모든 부분에 대한 정확한 상태와 제어하는 작업에 이용

－자료관리

자료들에 대한 처리규칙 등을 기록하고 관리, 유지하여 정보의 무결성, 논리적 정보구조, 정보의 구성 부분을 정의하는데 이용

－데이타베이스 관리

DBMS에서 제공하는 모든 정보를 이용하여

모델링, 무결성을 위한 데이타베이스 조정, 사용자 접근 제어, 회복과 재저장, 데이타의 효과 평가등에 이용

－문서관리

IRDS는 구조, 구현 전략, 처리 환경, 사용자, 제어, 보존(Retention) 등을 정의하여 관리, 형식화, 사용, 문서 감독 등을 수행

－통신망 관리

각 통신장비의 능력과 상호 연결 관계, 접근 가능 여부, 한 장비가 고장일때 다른 접근경로의 결정등을 기록하고 유지

－프로젝트 관리

프로젝트 관리에 필요한 제반 사항을 정의하여 정보들에 대한 정보 가용성, 정보의 형태, 다른 정보와의 연관성 등을 이용하여 프로젝트 관리에 필요한 정보를 제공

－시스템 공학

IRD내에 시스템 요구사항을 저장시켜 각 출력물과 연관된 상세 설계와 구조를 기술

－CASE 도구

정보저장소를 중심으로 통합하여 서로 다른 도구들 사이의 정보 공유 및 상호 연관성을 유지

4. IRDS에 기반한 CASE도구 정보저장소의 구현

충남대학교 컴퓨터과학과 데이타베이스 연구실에서는 국내에서 처음으로 IRDS의 개념을 이용하여 다양한 응용분야에 적용하기 위한 초기 단계로서 IRDS 명령어를 처리할 수 있는 IRDS 핵심 모듈인 서비스 인터페이스 및 그래픽 사용자 인터페이스, 응용프로그램이 IRD 및 IRD 스키마를 접근할 수 있는 수단을 제공하는 CLI(Call Level Interface) 형식의 인터페이스, 그리고 시스템 개발 전체 생명주기 관리 기능을 제공하는 통합 CASE 도구의 정보저장소로서 IRDS를 활용하기 위해 시스템 분석 및 설계 단계에 생성되는 산출물을 분석하여 IRDS 모델링을 통해 두 단계간의 상호참조 추적을 위한 스키마를 구현하였다.

4.1 IRDS 서비스/사용자 인터페이스 구현

IRDS를 실행시키면 IRDS 사용자 이름을 입력받아 저장하고, IRD들의 이름을 저장하고 있는 테이블의 존재여부를 결정해, 사용자 이름이 존재하지 않을 경우에는 IRD 생성기를 통해 DBMS의 도움을 받아 각 단계별로 테이블을 생성하며, 사용자 인터페이스를 통해 IRDS 명령어를 입력받아 문법검사와 일관성 검사를 한 후 데이타베이스 서비스 인터페이스를 통해 처리되도록 그림 4와 같은 구조를 갖는다.

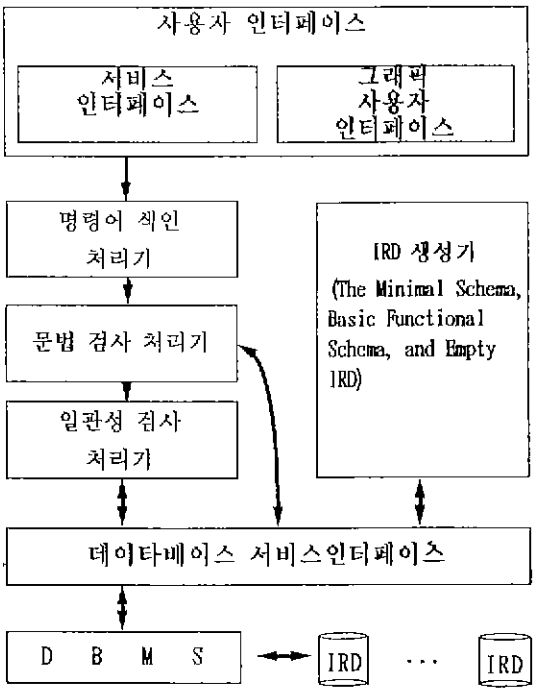


그림 3 IRDS의 전반적인 구조

또한 ANSI IRDS에서는 IRD 스키마 및 IRD 제어와 무결성 유지를 위해 Minimal 스키마와 Basic Functional 스키마를 구현 초기에 IRD에 저장하도록 하고 있다. Minimal 스키마는 IRDS의 근본적인 무결성 유지를 위한 데이타를 공유하기 위하여 정의되며, Basic Functional 스키마는 IRDS를 사용하는 조직체가 공동으로 사용할 수 있는 자료의 타입을 미리 정의하고 있다.

따라서 IRDS의 기본구조에 근거하여 사용되는 테이블은 다음과 같다.

－IRD 스키마 정의 단계 테이블

MAT(Meta-Attribute-Type)
MAGT_MAT(Meta-Attribute-Group-Type/Meta-Attribute-Type)
MET_MAGT(Meta-Entity-Type/Meta-Attribute-Group-Type)
MET_MAT(Meta-Entity-Type/Meta-Attribute-Type)
MRT_MAT(Meta-Relationship-Type/Meta-Attribute-Type)
－IRD 스키마 단계 테이블
MENT_MAT(Meta-Entity/Meta-Attribute)
MENT_MAG(Meta-Entity/Meta-Attribute-Group)
MREL_MATT(Meta-Relationship/Meta-Attribute)
－IRD 단계 테이블
ENTY_ATT(Entity/Attribute)
ENTY_AG(Entity/Attribute-Group)
REL_ATT(Relationship/Attribute)

• 사용자 인터페이스

IRDS 명령어에 익숙하지 않은 사용자를 위해 IRDS에 쉽게 접근할 수 있도록 표준적인 기능을 갖춘 그래픽 사용자 인터페이스를 제공한다. 이를 위해 IRD 내용을 조작하는 IRDS 명령어를 IRD 명령어, IRD 스키마 명령어, 유틸리티 명령어로 각각 구분하여 OSF/Motif를 이용하여 구현하였다. 양질의 사용자 인터페이스를 제공하기 위해 모든 IRDS 명령어가 메뉴로 처리될 수 있도록 하향식 메뉴를 제공한다. 화면 구성은 IRDS 명령어를 선택할 수 있는 명령어 선택 영역, 각 명령어를 입력할 수 있는 명령어 입력 영역, IRDS 명령어 스트링을 IRDS 서비스 인터페이스에 전달하여 실행 결과를 출력하는 메시지 영역과 IRDS 명령어를 입력하는 과정중에 명령어를 모를 경우 명령어 문법을 보여주는 도움말 영역으로 나누었으며, 각 IRDS 명령어마다 명령이 문법이 다르기 때문에 IRDS 명령어 따른 입력화면을 따로 두어 구현하였다.

• IRD 생성기

사용자가 IRDS를 실행시키면, IRDS는 IRD가 저장된 테이블에서 사용 가능한 IRD가 있는지 조사한다. 만약 사용 가능한 IRD가 존재하지 않으면, IRDS는 필요한 테이블을 생성하기 위하여 먼저 IRD 스키마 정의 단계의 테이블들을 생성한 후, IRD 스키마 단계의 테이블들을 생성한다. IRD 스키마 정의 단계의 테이블들은 한번 생성되어 인스턴스가 채워지면 모든 IRD들에 의해 공통으로 사용되며, 사용자가 명시적으로 사용 가능한 IRD들을 삭제하지 않는한 영원히 존재하고 사용자가 이 테이블들을 수정할 수 없다. IRD 스키마 단계의 테이블들에는 IRD 스키마 정의 단계의 테이블에 저장되는 인스턴스를 저장하는 동시에 IRD 단계에 저장되는 정보에 대한 타입을 저장한다. 마지막으로 IRD 단계의 테이블들을 생성하며, IRD 스키마 단계에 저장되는 타입을 저장하는 동시에 응용단계에 저장되는 정보에 대한 타입을 저장한다. 사용자가 입력한 정보가 메타스키마에 대한 정보라면 IRD 단계와 IRD 스키마 단계의 테이블에도 정보가 수록된다.

• 명령어 색인 처리기

사용자 인터페이스를 통해 입력된 IRD 명령어와 IRD 스키마 명령어를 단어별로 분류하여 전역 변수에 저장하며, 입력된 IRDS 명령어의 종류를 파악하여 문법 검사 처리기에 넘겨준다.

• 문법 검사 처리기

명령어 색인 처리기가 알려주는 IRDS 명령어의 종류에 따라 Recursive-descent 파싱 방법을 이용하여 토큰을 하나씩 반복적으로 입력받아 해당 명령어별 명령어에 맞는 템플리트를 가지고 단어별로 문법 검사를 한다.

• 일관성 검사 처리기

사용자 인터페이스를 통하여 IRDS 명령어가 입력되고 문법 검사가 이루어진 후에 IRDS 명령어에 맞는 명령어의 의미와 입력된 식별자의 타당성을 검사하기 위해, Informix 동적 SQL중에 SELECT 명령어를 이용한다. 일관성 검사를 위한 필요한 데이타는 Informix DBMS가 관리하는 테이블로 구성되어 있기때문에 데

이타베이스 표준 질의어인 SQL로 전환되어야 한다. 이러한 일관성 검사는 IRDS 명령어들에 일반적으로 적용되기 때문에 IRDS에서 HLI로 시작하는 전용함수들로 구현하였다.

- 데이타베이스 서비스 인터페이스
- 사용자 인터페이스를 통하여 입력된 모든 IRDS 명령어는 결국 DBMS를 이용하여 필요한 작업을 수행한다. DBMS는 이러한 모든 명령어들이 접근하는 IRD들을 관리하고 필요한 서비스를 제공한다. 또한 데이타베이스 서비스 인터페이스는 IRD에 저장된 데이타의 접근과 조작을 필요로 하는 모든 프로세스에게 DBMS를 사용하기 위한 서비스 인터페이스를 제공하며, IRD 스키마 정의, IRD 스키마, 그리고 IRD 단계의 테이블에 저장된 데이타를 관리하는 수단을 제공한다. 이러한 데이타베이스 서비스를 제공하기 위해 Informix 관계형 DBMS를 하부구조를 갖도록 구현하였으며, IRDS에서 DBMS를 접근하기 위해 SQL 명령어를 사용하여 데이타베이스 서비스 인터페이스를 접근하는 ESQL/C 프로그램을 이용하였다. 또한 데이타베이스 서비스 인터페이스를 처리하는 과정에서 발생하는 오류를 처리하기 위하여 Informix 데이타 처리오류 회복을 위해 제공되는 SQLCA. SQLCODE 값을 리턴하여 개발자에게 오류 원인을 제공하도록 구현하였다.

4.2 IRDS C CLI의 구현

모든 IRDS의 명령어를 사용자 인터페이스를 통하여 입력받고 입력된 명령어들의 처리결과를 모두 사용자 인터페이스의 화면으로 출력한다. 이러한 명령어 입출력 방법은 CASE 도구의 정보저장소로서 IRDS를 사용하는데 불편함이 있다. 따라서 CASE 도구와 IRDS간의 정보저장소 인터페이스 역할을 하는 C CLI (Call Level Interface) 형식의 새로운 명령어 입출력 방법이 필요하다.

IRDS C CLI는 C 프로그램에서 IRDS의 서비스가 필요할 때 IRDS에게 IRDS 명령어를 전달하고 그 결과를 C 프로그램으로 돌려줄 수 있도록 하기 위한 고려사항으로는 다음과 같다.

- IRDS 정보저장소 인터페이스 방법
- IRDS 명령어 전달 기능
- IRDS 명령어 처리 결과 저장 기능
- IRDS 명령어 실행 판단 기능

4.2.1 IRDS 정보저장소 인터페이스 방법

IRDS 정보저장소 인터페이스 방식은 크게 두가지로 분류할 수 있다[11].

첫번째 방법은 주 언어에서 호출하여 연결기(Linker)에 의해 실행시간에 연결되는 라이브러리 함수들의 인수로 IRDS 명령어를 전달하는 방법이다. 이 방법은 IRDS 명령어의 처리과정이 IRDS 명령어를 C 프로그램에 삽입하여 선행 처리 과정을 거치는 것보다 훨씬 간결해지므로 구현하기가 비교적 쉽다. 이러한 인터페이스 방법의 대표적인 예로는 Oracle 관계형 DBMS에서 제공하는 OCI(Oracle Call Interface)가 있다. 두번째 방법은 주 언어에 IRDS 명령어를 삽입(Embedded)하여 선행처리기에 의해 주 언어에 맞도록 원시 코드를 선행처리한 후 이를 다시 주 언어 컴파일러를 이용하여 실행 코드를 생성하는 방법이다. 이 방법의 예로서는 DBMS의 한 모듈로서 제공되는 ESQL/C 선행처리기를 이용하여 주 언어인 C 프로그램에 SQL을 삽입하여 응용프로그램을 작성하는 방법이다.

본 연구팀에서는에서는 IRDS 선행처리기를 구현하기 위해 반드시 필요한 첫번째 방법을 따르는 IRDS C CLI를 구현하였다.

4.2.2 IRDS 명령어 전달 기능

IRDS 명령어를 가리키는 문자열 포인터 변수를 IRDS C CLI 라이브러리 함수의 인수로 전달하여 IRDS에게 명령어를 전달할 수 있도록 하였다. 이러한 문자열 변수는 주 언어인 C로 작성되는 프로그램에서 개발자에 의해 선언되고, 선언된 문자열 포인터 변수에 IRDS 명령어가 정적 또는 동적으로 바인딩되어 IRDS C CLI를 구성하는 라이브러리 함수를 호출할 때 인수로 전달된다. 이러한 인수는 IRDS C CLI를 구성하는 라이브러리 함수에서 'irdscmd'라는 문자열 포인터 변수에 저장된다. 문자열을 가리키는 포인터 변수는 C 언어에서 제공하는 표준 라이브러리 함수들 중에서 문자열을 관리

하는 라이브러리 함수로 전달할 수 있으므로 IRDS 명령어의 전달 기능을 효율적으로 처리해 줄 수 있다. 또한 CASE 도구와 결합하여 하나의 프로세스로 사용할 때 간단하게 이용할 수 있다.

4.2.3 IRDS 명령어 처리 결과 저장 기능

IRDS C CLI를 통해 전달되는 IRDS 명령어들 중에서 IRD에 저장된 메타 정보와 메타 스키마 정보를 검색하는 명령어는 'output ird' 명령어와 'output ird-schema' 명령어가 있다. 이외의 IRDS 명령어는 IRD에 메타 정보나 메타 스키마 정보를 삽입, 삭제, 수정하는 명령어들과 새로운 IRD를 생성하거나 삭제하는 명령어들이다. 따라서 'output ird' 명령어와 'output ird-schema' 명령어를 제외한 명령어들은 검색 결과를 저장하기 위한 특별한 자료구조가 필요하지 않다. 그러므로 IRD 명령어 처리 결과를 주 언어에서 처리할 수 있도록 하기 위한 메타 정보와 메타 스키마의 구조는 그림 5와 같은 구조체형을 갖는 변수를 선언한 후 'outptut ird' 명령어나 'output ird-schema' 명령어를 IRDS C CLI를 통해 IRDS에게 전달한다.

IRDS C CLI는 IRDS에 의해 처리된 검색

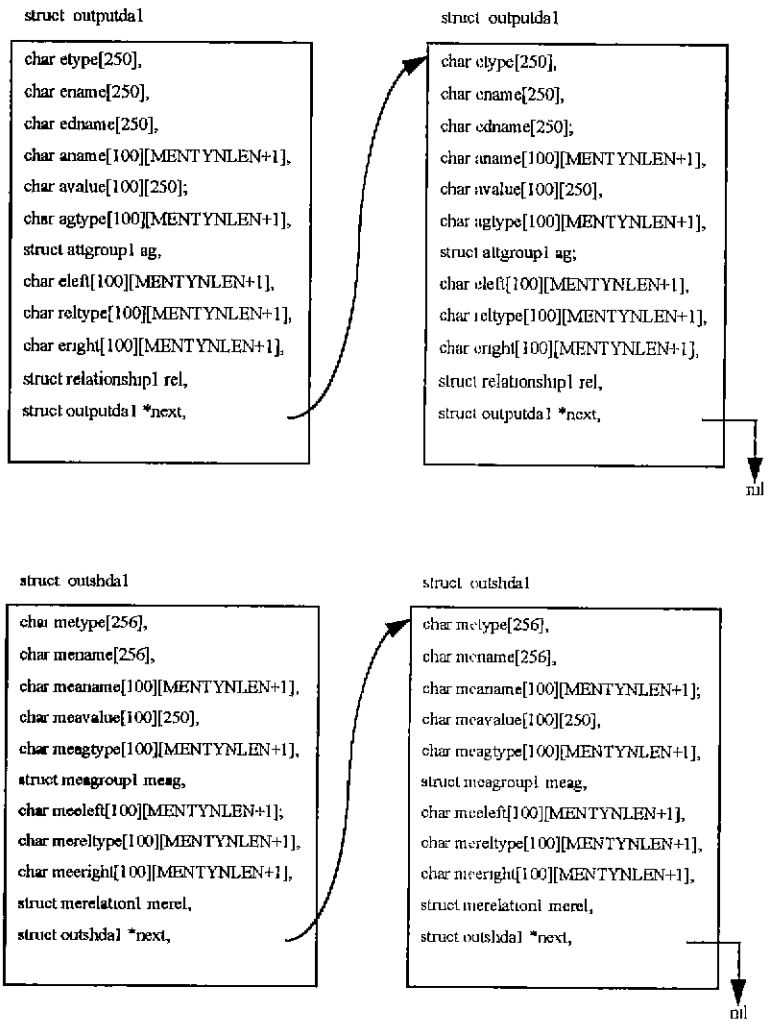


그림 4 'output ird'와 'output ird-schma' 명령에 의한 처리결과 저장 구조

결과를 개발자에 의해 선언된 검색 결과 저장을 위한 구조체를 가리키는 변수에 저장한다. 개발자는 이 변수를 사용하여 검색 결과를 주 언어상에서 원하는 대로 처리한다.

4.2.4 IRDS 명령어 실행 전달 기능

IRDS C CLI를 통해 전달된 IRDS 명령어가 IRDS에 의해 성공적으로 실행완료 되었는지, 혹은 오류가 발생했는지를 파악하기 위하여 실행 판단 변수인 'irdscode'에 저장한다. 전역변수 'irdscode'값에 따라 IRDS 명령어 실행 판단을 할 수 있다. 다음은 'irdscode'에 저장되는 값의 의미를 나타낸다.

- IRDS 오류

문법 오류 : −5000~ −6000
명령어 의미 오류 : −7000~ −8000
실행시간 오류 : −8000~ −9000

- 한글 Informix ESQL/C 오류

−100~ −1300

- 성공적인 명령어 실행

0 보다 큰값

4.3 CASE 정보저장소를 위한 IRDS 모델링

정보저장소의 메타모델은 정보저장소 전체 내용에 대한 전반적이고 개념적인 관점에 입각하여 정보저장소에 기초를 둔 CASE 도구들이 어떤 소프트웨어 생명주기와 방법론을 지원하는지를 결정한다. 즉 메타모델은 정보저장소에 저장될 정보의 종류, 이들 정보의 형태와 성질, 이들 정보에 접근하고 정확성을 증명하는 방법 등을 정의한다. 만일 서로 다른 생명주기 단계에서 다른 방법론을 사용한다면, 단계간에 정보를 잃거나 중복되지 않도록 조정이 필요하며, IRDS에서는 IRD 스키마 정의를 통해 이러한 조정이 가능하다[11, 14, 15, 19].

메타모델이 표현되는 형태는 크게 E-R 모델과 객체지향 모델이 있다. ANSI IRDS는 E-R 모델로 표현되며, IRDS 구조상 메타모델은 IRD 스키마에 해당하고 실제 저장 정보는 IRD 메타데이타에 해당한다[1, 14, 19].

따라서 구조적 방법론을 따르는 소프트웨어 개발과정 중에서 시스템 요구분석 및 설계 단계에서 생성되는 산출물을 대상으로 정보저장소 메타모델을 설계하여 IRD 스키마로 전환하고, 이들의 자료 종류와 그들간의 관계를 파악하여 IRD 스키마에서 사용하는 E-R 모델로 변환이 필요하다.

4.3.1 IRD 스키마를 위한 IRDS 모델링

스키마는 데이타베이스나 자료사전에 있는 정보를 나타내기 위한 수단과 정보의 타입을 결정한다. 메타데이타 타입은 자료사전을 이용하기 전에 정의되어야 한다. IRDS는 기존에 이미 정의되어 있는 여러가지 메타데이타 타입을 Minimal 스키마와 Basic Functional 스키마를 통해 제공하고 있다. IRDS 사용자는 특정 응용을 위해 필요한 부가적인 메타데이타 타입을 정의하여 확장된 스키마를 사용할 수 있으며, 이때 이러한 메타데이타 타입은 각 IRD 스키마에 적절히 명시되어야 한다. IRD 스키마 구조는 사용자가 IRD에 나타낼 수 있는 정보의 타입을 결정한다[19].

따라서 기존의 정보저장소 메타모델로 부터 새롭게 IRDS에 적합한 모델로 재구성하기 위해서는 메타모델에서 지원하고 있는 정보의 유형-예를 들면, 프로젝트, 자료흐름도, 프로세스 명세, 결정표, 구조도 등에 따라 정의할 객체의 종류, 객체간의 관계, 객체의 속성 등을 분석해야 한다.

4.3.1.1 생명주기 단계의 모듈을 위한 스키마

조직에서 생명주기 개발 정보를 파악하기 위해 IRDS를 이용하는 경우, 사용자는 하나의 IRD를 여러 단계로 분할하여야 한다. IRD를 생명주기 단계와 관련시켜 분할할 때 각 단계와 관련된 스키마는 한번에 또는 점진적으로 정의될 수 있다. 또한 분할된 단계는 독립적인 하나의 스키마로 정의되거나 다른 스키마와 관련시켜 정의될 수 있다[1, 19].

따라서 IRD 스키마 정의 과정에서 사용자는 IRDS 생명주기 단계 모듈을 정의함으로써 동일 생명주기 단계 내 또는 서로 다른 단계간의 상호참조 및 무결성을 지원할 수 있다.

핵심 IRDS에서는 사용자가 생명주기 단계에 상응하는 IRD-Partition을 구성할 수 있도록 기본적인 생명주기 단계로서 요구사항, 설

계, 개발 등과 같이 시스템 생명주기에서 non-operational 단계를 나타내는 Uncontrolled 단계가 있으므로 시스템 요구분석 및 설계 단계에 따른 IRD-Partition을 다음과 같이 정의할 수 있다.

Add meta-entity SYSTEM_ANALYSIS
meta-entity-type = IRD-Partition ;
Add meta-entity SYSTEM_DESIGN
meta-ntity-type = IRD-Partition ;

4.3.1.2 IRD 스키마를 위한 E-R 모델링

IRDS 모델링의 대상이 되는 주제에 대한 일반적인 E-R 모델이 IRDS 모델을 염두에 둔 모델이 아닌경우 곧바로 IRDS 모델로 전환하기가 어렵다. 따라서 메타모델과 IRDS 모델에서 개체형과 관계형이 직접 일대일 관계로 대응되지 않을 수 있기 때문에 고려할 사항으로는 첫째, 반드시 정의되어야 할 개체형 및 관계형을 추출하고, 둘째, 일관성 있는 이름부여가 선행되어야 하며, 세째, 각 단계의 개체들을 연관시키는 관계형을 정의하여야 하며, 네째, 이미 정의된 스키마의 활용 가능성과 부가적인 스키마를 정의하여 사용할 것인가를 고찰하며, 다섯째, 개체간의 관계를 정의한 관계형이 양방향 관계가 필요한 관계형은 반드시 두 개체형에 대해 두개의 관계형으로 따로 정의하여야 할 필요가 있다.

• 개체형 정의

메타모델로 부터 IRD 스키마를 위한 개체형을 추출하는 과정은 기존의 메타모델의 개체를 그대로 IRD의 개체형으로 대응시킴으로써 가

표 1 메타모델과 IRD 스키마의 개체형 비교

단 계	메타모델의 개체 이름	IRD 스키마의 개체형 이름	Alternative Name
시스템 요구분석 단계	DATA FLOW DIAGRAM	P-DFD	Project-DFD
	DIAGRAM OF DFD	DFD	DFD
	DFD OBJECT	PROCESS	Process
		DATA-STORE	Data-Store
		EX-ENTITY	External-Entity
	DFD DATA	DFD-DATA	Data-Flow
	ATTRIBUTE	ATTRIBUTE	Data-Dictionary
	PROCESS SPEC	PROCESS-SPEC	Mini-spec
시스템 설계 단계	STRUCTURE CHART	P-SC	Project-SC
	DIAGRAM OF SC	SC	Structure-Chart
	SC OBJECT	ROOT-MODULE	Root-module
		STAND-MODULE	Standard-Module
		TRANS-MODULE	Transaction-Module
		LIB-MODULE	Library-Module
		DB-MODULE	Database-Module
	SC DATA	SC-DATA	Structure-Data
	PACKAGE	PACKAGE	Package
	MODULE SPEC	MODULE-SPEC	Mini-spec

능하다. 또한 각각의 개체가 공통적으로 갖는 속성형 및 관계형이 있는 동시에 각 개체가 서로 다른 속성형 및 관계형을 갖고 있으므로, IRDS 모델을 위해서는 확연히 구분된 개체로 정의되어야 하며, 개체형과 독립적으로 관계형 및 속성형을 일관성 있게 정의한 후 이를 이미 정의된 개체형과 연관시켜야 한다.

표 1은 시스템 요구분석 및 설계 단계의 메타모델로 부터 IRD 스키마를 위한 개체형이 어떻게 대응되는가를 보이고 있다.

• 속성형 정의

개체의 특성을 기술하는 정보는 속성형으로 정의된다. 이미 정의된 개체형이 갖을 수 있는 모든 속성을 분류하고, 이 가운데 공통적인 속성들을 추출하여 속성형으로 정의할 수 있다. 핵심 IRDS 스키마에 이미 정의되어 있는 속성형은 모든 시스템에서 사용될 수 있는 기본적인 정보를 제공하므로 그대로 이용하고 여기에 각 생명주기 단계에 필요한 부가적인 속성형만을 정의하여 나타낸 예는 표 2와 같다.

• 관계형 정의

자료흐름도 및 구조도의 메타모델에 정의된 관계형들은 이미 IRD 스키마로 정의된 개체간의 상호연관성을 고려하여 이에 적합한 관계형으로 표현하여야 한다. IRD 스키마를 위한 관계형은 크게 소프트웨어 생명주기 단계의 관점에서 동일 단계 정보 무결성을 위한 관계형과 서로다른 단계 정보 무결성을 위한 관계로 나누어 볼 수 있으며, 다시 이를 개체간의 관계 자체의 의미에 따라 관계클래스형을 정의할 수 있다. 관계클래스형은 두 개체형을 특정한 의미로 묶어주는 역할을 하므로 어떤 관계 인가에 따라 표 3과 같이 분류할 수 있다. 이러한 관계클래스형을 통해 여러 개체쌍이 연결됨으로써 관계형을 이룬다.

4.3.1.3 IRDS 모델

위에서 정의된 개체형, 관계형, 속성형을 이

표 2 개체형과 속성형의 결합 예

단 계	개 체 형	속 성 형	정의 주체	속성 반복	비 고
	모든 개체형에 공통적인 속성형	ADDED-BY LAST-MODIFIED-BY MUNBER-OF-TIME-MODIFIED	M	S	감사를 위해 시스템에 의해 자동으로 관리 됨
		COMMENTS	B	S	FMT : TEXT(240)
		DESCRIPTION	B	S	FMT : TEXT(5000)
		DESCRIPTIVE-NAME	B	P	SING : NO
시스템 요구분석 단계	DFD	CONTEXT-DFD	U	S	ATTRIBUTE-TYPE-VALIDATION-DATA (YES-OR-NO-VALUE)를 이용함
		DFD-NUMBER	U	S	SYS-GENED = YES SYS-MAINT = YES UNIQUESS-RULE = YES
시스템 설계단계	SC	ROOT-SC	U	S	ATTRIBUTE-TYPE-VALIDATION-DATA (YES-OR-NO-VALUE)를 이용함

. 정의주체(Minimal 스키마 : M, Basic Functional 스키마 : B, 사용자 정의 : U)

. 속성의 반복(Singular : S, Plural : P)

표 3 관계클래스형에 따른 관계형의 분류

단계	관계분류	관계클래스형	관련된 개체형쌍의 예	관계형 정의 예
동일 단계내 관계형	인터페이스	COMES-FROM, GOES-TO (TARGET-OF, SOURCE-OF) (ACCESS, ACCESSED-BY)	(DFD-DATA, PROCESS) (PROCESS, PROCESS) (PROCESS, DATA-STORE)	DFD-DAAT-COMES-FROM-PROCESS PROCESS-SOURCE-OF-PROCESS PROCESS-ACCESS-DATA-STORE
	순환 계층관계	(PARENT-OF, CHILD-OF)	(DFD, DFD)	DFD-PARENT-OF-DFD
	구성관계	CONTAINS	(DFD, PROCESS)	DFD-CONTAINS-PROCESS
	다른 개체형으로서의 분기관계	EXPLODES-TO	(PROCESS, DFD)	DFD-EXPLODES-TO-DFD
	상세기술	(SPECIFIES)	[PROCESS,	PROCESS-SPECIFIES-PROCESS-SPE
	관계		PROCESS-SPEC]	C
단계간 관계형	상호참조 관계	REFER-TO	(DFD, SC)	DFD-REFER-TO-SC

( ) : 양방향 관계형, [ ] : 개체형 쌍

용하여 시스템 요구분석 및 설계 단계의 메타모델을 IRD 스키마를 위한 E-R 모델로 나타낼수 있다. 그림 6은 시스템 요구분석 단계의 메타모델에 적용한 예이고, 그림 7은 시스템 요구분석 및 설계 단계의 각 단계간 상호참조를 가능케하는 스키마를 보인 예이다.

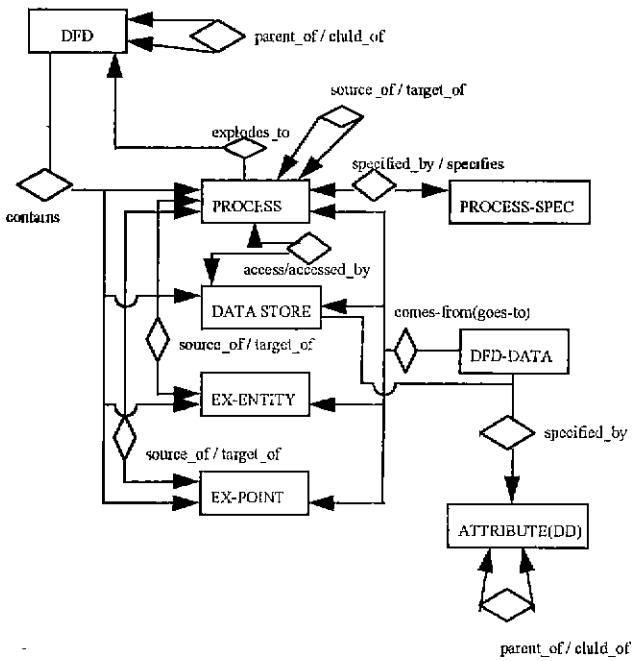


그림 5 자료흐름도를 위한 IRD 스키마 모델

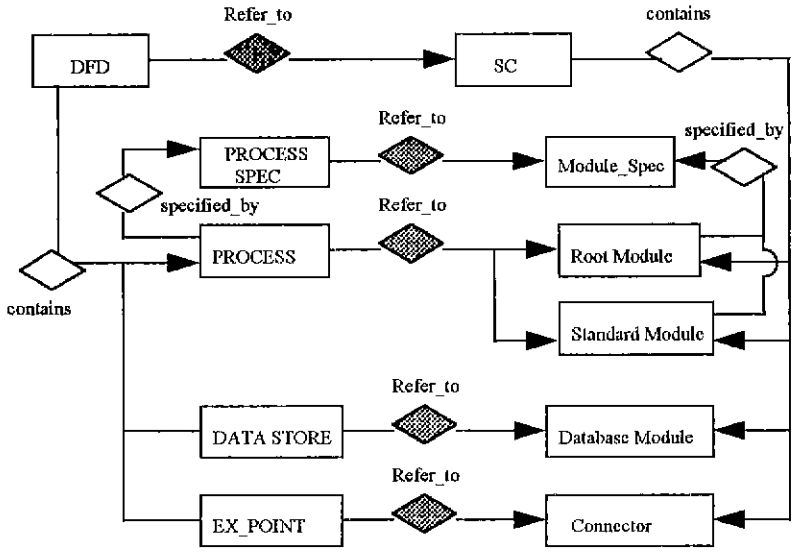


그림 6 자료흐름도와 구조도간의 상호참조를 위한 IRD 스키마 모델

4.3.2 IRD 스키마 정의

일단 E-R 모델이 완성되면 사용자는 IRD 응용에 적합한 생명주기 단계, 뷰 그리고 스키마 구조를 정의할 수 있다. 이미 IRDS에 정의된 스키마만을 이용한 경우에는 IRD 스키마를 별도로 정의할 필요없이 메타데이타를 곧바로 입력할 수 있으나, 사용자가 확장한 스키마를 이용하기 위해서는 IRDS 명령어를 이용하여 새로운 스키마를 정의하여 메타데이타를 입력하여야 한다.

따라서 IRD 스키마를 위한 E-R 모델을 IRDS 명령어에 따라 IRD 스키마를 정의한 예는 표 4와 같다.

4.3.3 IRDS 모델의 적용

앞에서 정의된 IRD 모델에 근거하여 구체적으로 CASE 도구의 정보저장소를 IRD 메타데이타로 적용하기 위해, 도서관리시스템 이란 특정 시스템을 저장 정보의 예로 선택하였으며, 이 시스템의 분석 결과인 자료흐름도 및 구조도를 중심으로 적용하였다.

시스템 요구분석 및 설계 단계의 IRD 스키마 모델에서 보듯이 각각 대상으로 한 자료흐름도 및 구조도의 내용이 실제로 IRD 메타데이타로 저장되기 위해서는 이를 객체, 속성, 관계로 등록되어야 하며, 표 5는 이를 IRDS 명령어를 이용하여 나타낸 예이다.

한편 IRDS 모델링 및 저장 정보의 실험 결과를 보기위해 메타데이타가 IRD 스키마에 따라 제대로 등록되었는가를 확인하기 위한 예로서 DFD 개체인 도서관리시스템-DFD-1이 갖는 모든 관계들을 검색 하는 명령어는 다음과 같다.

output ird select entites access-name = 도서관리시스템-DFD-1 show all relationships ;

수행결과

Entity = 도서관리시스템_DFD_1
Entity Descriptive-NAme = 도서관리시스템 Data_Flow_Diagram_No_1
Entity Type = DFD
Relationships
도서관리시스템_DFD_1 DFD_CONTAINS_ DATA_STORE 도서목록_DATA_STORE
도서관리시스템_DFD_1 DFD_CONTAINS_ EX_ENTITY 이용자
도서관리시스템_DFD_1 DFD_CONTAINS_ PROCESS 도서등록_PROC
도서관리시스템_DFD_1 DFD CHILD_OF _DFD 도서관리시스템_DFD_0
도서관리시스템_DFD_1 DFD_REFER-TO

표 4 시스템 요구 분석 및 설계 단계와 단계간 상호참조를 위한 IRD 스키마 정의 예

	시스템 요구분석 단계	시스템 설계 단계	단계간 상호참조
생명주기 단	Add meta-nelity SYSTEM -ANALSIS meta-entity-type = IRD-PARTITION	Add meta-entity SYSTEM -DESIGN meta-entity-type = IRD-PARTITION	
개체형 정의	Add meta-entity DFD meta-entity-type ENTITY-TYPE	Add meta-nelity SC meta-entity-type = ENTITY-TYPE	
속성형 정의	Add meta-entity CONTEXT-DFD meta-entity-type = ATTRIBUTE-TYPE with SYS-GENED = 'YES', PURPOSE = 'CONTEXT 다이어그램 여부'	Add meta-entity ROOT-SC meta-entity-type = ATTRIBUTE-TYPE with SYS-GENED = 'YES', PURPOSE = 'ROOT 다이어그램 여부'	
관계형 정의	Add meta-entity DFD-CONTAINS-PROCESS meta-entity-type = RELATIONSHIP-TYPE	Add meta-entity ROOT-MODULE-CALLS-STAND-MODULE meta-entity-type = RELATIONSHIP-TYPE	Add meta-entity DFD-REFER-to-SC meta-entity-type = RELATIONSHIP-TYPE
개체형과 속성형의 결합 정의	Add meta-relationship DFD entity-type-contains-attribute-type CONTEXT-DFD	Add meta-relationship SC entity-type-contains-attribute-type ROOT-SC	
관계클래스 정의	Add meta-entity CONTAINS meta-entity-type = RELATIONSHIP-CLASS-TYPE	Add meta-entity CONNECT-TO meta-entity-type = RELATIONSHIP-CLASS-TYPE	Add meta-entity REFER-TO meta-entity-type = RELATIONSHIP-CLASS-TYPE
관계형과 관계클래스형의 결합 정의	Add meta-relationship DFD-CONTAINS-PROCESS relationship-type-mamber-of-relationship-class-type CONTAINS	Add meta-relationship SC-CONTAINS-ROOT-MODULE relationship-type-mamber-of-realtionship-class-type CONTAINS	Add meta-relationship DFD-REFER-TO-SC relationship-type-member-of-relationship-class-type -relationship-class-type REFER-TO
관계형 위치 정의	Add meta-relationship DFD-CONTAINS-PROCES relationship-type-connects-entity-type DFD position = 1 Add meta-relationship DFD-CONTAINS-PROCES relationship-type-connects-entity-type PROCESS position = 2	Add meta-relationship SC-CONTAINS-ROOT-MODULE relationship-type-connects-entity-type SC position = 1 Add meta-relationship SC-CONTAINS-ROOT-MODULE relationship-type-connects-entity-type ROOT-MODULE position = 2	Add meta-relationship DFD-REFER-TO-SC relationship-type-connects-on tity-type DFD position = 1 Add meta-relationship DFD-REFER-TO-SC relationship-type-connects-en tity-type SC position = 2

_SC 도서관리시스템_SC

위에서 적용한 IRDS 명령어는 특정 개체명과 관련이 있는 모든 관계 및 관련된 개체를 출력하고자 한 것이다. 이 명령어의 출력 결과는 크게 두부분으로 나뉘어 진다. 첫째 부분은 해당 개체명의 기본적인 정보로써 개체명, 개체의 기술명 및 개체형을 나타내고, 둘째 부분은 개체명과 관련된 모든 관계들을 나타낸다.

특히 시스템 요구분석 단계의 개체형인 DFD가 시스템 설계 단계의 개체형인 SC와 상호참

표 5 시스템 요구분석 및 설계 단계의 메타데이타 정의 예

	시스템 요구분석 단계	시스템 설계 단계
개체 정의	Add entity 도서등록-PROC entity-type = PROCESS	Add entity 도서등록-MDL-SPEC entity-type = MODULE-SPEC
관계 정의	Add relationship 도서등록-PROC PROCESS-SPECFIED-BY-PROCES S-SPEC 도서등록-PROC-SPEC	Add relationship 도서등록-SC SC-CONTAINS-LIB-MODULE 입력자료 확인-MDL
자료흐름도 및 - 구조도 간의 상호참조 관계 정의	Add relationship 도서등록-PROC PROCESS-REFER-TO- STANDARD-MODULE 도서등록-MDL	

조 관계(REFER_TO)를 갖고 있음을 보이고 있다. 따라서 서로 다른 단계간의 상호참조 추적을 위한 스키마에 해당하는 관계형인 DFD_REFER_TO_SC가 실제 메타데이타에서 상호참조가 이루어 짐을 확인할 수 있다.

5. 결론 및 향후 연구과제

CASE 도구의 정보저장소로서 IRDS를 이용하기 위해서 ANSI IRDS 표준안에 따라 IRDS 서비스 및 사용자 인터페이스를 구현하였으며, 이를 응용프로그램에서 활용하기 위한 방법으로서 C 언어와의 정보저장소 인터페이스인 CLI를 구현하였으며, 구조적 방법을 따르는 소프트웨어 개발과정 중에서 시스템 요구분석 및 설계 단계에서 생성되는 산출물을 대상으로 정보저장소 메타모델을 정의하고 실제 예인 도서관리시스템을 통해 두 단계간의 상호참조 추적이 가능함을 보였다.

IRDS가 CASE 도구의 정보저장소로서 제 기능을 다하기 위해서는 현재 단일 사용자 환경에서 수행되는 IRDS를 다중 사용자 환경에서 수행될 수 있도록 다자간 프로세스 통신 기능이 필요하다. 또한 소프트웨어 생명주기 각 단계 모두를 하나의 통합된 IRD 스키마로 정의하여 소프트웨어 생명주기 마다 다른 CASE 도구들이 지원되더라도 각 도구들이 하나의 정보저장소를 이용함으로써 통합된 CASE 도구 환경의 구축이 필요하며, 시스템 요구분석 및 설계 단계 뿐만아니라 전체 생명주기를 지원할 있도록 시스템 확장이 필요하다.

참고문헌

[1] Alan Goldfine, Patricia Konig, "A Technical Overview of the Information Resource Dictionary System(Second Edition)", U. S. Department of Commerce, January 1988.

[2] ANSI X3H4. 1, The Information Resource Dictionary System(IRDS) Reference Model, X3H4/90-195R, May 1991.

[3] Daniel R. Dolk, et al "Relational Information Resource Dictionary System", CACM, Vol30, No 1, January 1987.

[4] Margaret Henderson Law, "Guide to Information Resource Dictionary System Applications : General Concepts and Strategic Systems Planning", NBS Special Publication 500-152, April 1988.

[5] Mark R. Jones, "Brave New World : A Vision of IRDS", Database Programming and Design, November 1991.

[6] Mark R. Jones, "Unveiling Repository Technology", Database Programming and Design, April 1992.

[7] Minder Chen, Ronald J. Norman, "A Framework for Integrated CASE", IEEE Software, March 1992.

[8] Mohan Prabandham, et al, "A View of the IRDS Reference Model", Database Programming and Design, March 1990.

[9] Robert E. Hodges, Information Resource Dictionary Services Architecture Technical Report, X3H4/93-048R1, June 1993.

[10] Dana M. Marks, "An Alternative to the

IRDS Services Interface", IBM Proposal to X3H4.

[11] 강병도, 우치수, "CASE 환경을 지원하는 정보저장소(Repository)의 구성요건", 정보과학회지 12권 2호, 1994.

[12] 박중기, 진성일, "IRDS C CLI의 설계 및 구현", 한국정보처리학회 논문집 2권 4호지, 1995.

[13] 시스템공학연구소, "소프트웨어 테스팅 도구 개발", 1993.

[14] 시스템공학연구소, "국산 주전산기용 소프트웨어 개발 지원도구 개발, 소프트웨어 정보저장소 개발", 과학기술처, 1994.

[15] 양해술, "I-CASE의 도구 통합과 방법", 정보과학회지 12권 2호, 1994. 3.

[16] 정경업, "정보자원사전시스템의 설계 및구현", 석사학위논문, 충남대학교, 1993.

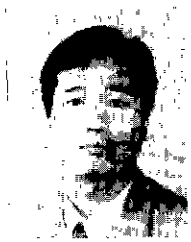
[17] 시스템공학연구소, 정보자원 사전 시스템의 설계 및 구현 , 1994.

[18] 이혜란, 최광준, 진성일, 김창갑, 신규상, CASE 도구의 정보저장소로서의 IRDS 활용 , 정보과학회 가을 학술발표논문집, Vol. 21, No. 2, pp 733-736, 1994.

진 성 일

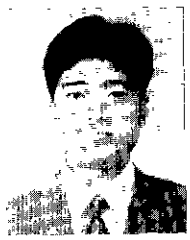
1978 서울대학교 계산통계학과 학사
1980 한국과학기술원 전산학과 석사
1994 한국과학기술원 전산학과 박사
1983~1989 충남대학교 전산학과 조교수
1987~1989 Northwestern 대학 전산학과 객원교수
1990~1992 충남대학교 전자계산소장
1989~1994 충남대학교 전산학과 부교수
1994~현재 충남대학교 전산학과 교수
관심분야 : 데이타베이스, 성능분석, 정보모델링, 멀티미디어

김 기 봉



1991 충남대학교 전산학과 학사
1993 충남대학교 전산학과 석사
1993~1994 충남대학교 자연과학대학 연구조교
1993~현재 충남대학교 전산학과 박사과정 재학중
1994~현재 혜전전문대학 전산정보처리학과 전임강사
관심분야 : 데이타베이스, 성능분석, 정보모델링

박 중 기



1993 한남대학교 전산학과 학사
1995 충남대학교 전산학과 석사
1995~현재 시스템공학 연구소
관심분야 : 주메모리 데이타베이스, 소프트웨어 공학

이 혜 란

1985 서강대학교 사회학과 강사
1995 충남대학교 전산학과 석사과정
1995~현재 시스템공학 연구소 국이공학 센타
관심분야 : 데이타베이스, 정보모델링, 자연이처리

조 유 희

1994 충남대학교 전산학과 학사
1994~현재 충남대학교 전산학과 석사과정 재학중
관심분야 : 데이타베이스, 정보모델링, 멀티미디어