□ 기술해설 □

CASE 환경을 지원하는 정보저장소 (Repository)의 구성 요건

서울대학교 강병도* · 우치수*

목 차

1. 서 론

소프트웨어 개발시 자동화된 도구들을 집적시킨 소프트웨어 개발환경을 구축하여 방대하고 복잡한 소프트웨어 개발과정을 신뢰성있게 지원할수 있다. 소프트웨어 개발환경내의 여러가지 도구들은 수작업으로 행할 행위들을 컴퓨터의 도움을 받아 빠른 시간내에 일관성있게 처리하여 줌으로써 생산성과 소프트웨어 품질을 향상시키는 데 기여하고 있다. 이러한 사실은 소프트웨어개발에 상당한 도움을 주었다. 예정된 기간내에 적정한 예산범위내에서 질높은 소프트웨어를 생산하는 것은 소프트웨어분야에 종사하는 대다수 사람들의 소망이었고, 상당히 어려운 작업이었다. 그래서, 1980년대부터 이러한 소망을 이룩하기 위하여 소프트웨어 개발과정을 자동화시키기 위한 많은 CASE(Computer-Aided Software Engineering)도구[1]를 개발하였다. 이 CASE 도구들은 소프트웨어 개발과정의 각 단계에 사용되어 수작업을 대신하여 신속하고 신뢰성있게 관련된 작업을 수행하였다. 이 도구들은 작동중에 여러가지 종류의 정보를 각각 사용하거나 새로이 생성하기도 한다. 따라서, 여기에서 새로이 제시되는 문제점이 여러종류의 도구들이 다루고 생성하는 정보들을 저장하고 관리하는 방법이다. 각각의 도구들이 중복되는 정보를 생성하기도 하고, 도구들간의 상호연관성이 적어서, 독자적인 작업을 수행하는 한계를 벗어나 도구들간의 정보공유로 인한 효율성의 극대화를 이루지 못하였다. 단위도구의 한계를 벗어나 각 도구들이 유기적인 연관관계를 맺어서 전

* 종신회원

체도구들이 가지고있는 기능을 극대화 시키는 방법은 도구들을 통합하는 것이다.

도구들을 통합하는 방법은 현재 크게 세가지로 나누어볼 수 있다. 첫번째 방법이 CASE 정보저장소(Repository)를 두어서 여러 도구들이 생성하고 사용하는 정보를 일괄 관리하는 것이다. 각 도구들이 생성하는 정보를 공유하게 함으로써 중복되는 정보나 일관성 없는 정보를 사전에 방지할 수 있다. 한 도구가 생성한 정보를 다른 도구가 이용하게 함으로써 소프트웨어 개발단계를 지원하는 도구들의 유기적인 결합을 유도할 수있다. 두번째 방법이 메세지교환을 통한 도구들의 결합이다[11]. 중앙에 정보저장소를 두는 것은 복잡하고 융통성이 낮으므로 중앙에 메세지 서버를 두어서 메세지를 통하여 도구들 상호간의 정보교환을 지원하는 것이 간단하고 융통성이 높다는 접근 방식이다. 세번째 방법은 링크를 통하여 상호관계되는 정보를 연결하자는 접근 방식이다[9].

본 고에서는 CASE 정보저장소 개념과 정보저장소를 통한 각 CASE 도구들의 통합방식에 관하여 언급하고자 한다. 그동안 정보저장소를 통한 도구들의 통합방식에 관해 많은 연구가 진행되어 왔다. 특정 프로그래밍언어를 지원하는 프로그래밍 지원 환경내의 도구들을 통합[8]하는 연구와 공통 서비스영역[7]을 두어서 도구들을 통합하는 접근 방식들을 사용하였다. 특정 프로그래밍언어를 지원하는 프로그래밍 환경내의 도구들만을 통합하는 방식은 개발단계 전체를 지원하지 못한다는 단점이 있다. 또한 어떤 접근 방식을 사용하느냐에 관계없이 공통적인 연구과제는 정보저장소 구조의 확장성이다. 즉, 새로운 도구가 CASE 환경내에 추가될 때 새로이 생성되는 정보를 기존의 정보저장소 메타모델에 어떤방식으로 충돌없이 저장하느냐 하는 문제가 커다란 연구과제 이다.

2절에서는 정보저장소의 주요개념에 관하여 살펴보고, 3절에서는 정보저장소의 정보관리 기능과 구조에 관하여 설명하고자 한다. 그리고, 4절에서는 정보저장소의 표준화 동향에 관하여 간략하게 소개하겠다.

2. 정보저장소

정보저장소는 소프트웨어 개발의 각 단계에서 사용되고 생산되어지는 모든 정보와 시스템 요소들을 저장함으로써 계획에서부터 유지 보수에 이르기까지 소프트웨어 생명 주기 전체를 지원한다. 이는 조직, 조직의 구조, 기업 모델, 프로시쥬어, 데이타 모델, 데이타 객체, 객체의 관련성, 프로세스 모델, 메타 모델, 실제적인 코드, 프로젝트 관리 정보, 각종 문서 등에 관한 정보를 저장하는 지식 베이스로 사용된다. 또한, 정보저장소는 이러한 많은 형태의 정보 뿐만 아니라, 이러한 정보들의 상호관계와 이들 정보들의 유효성을 증명하고 사용하는 규칙도 포함하여, 시스템의 이해와 변화 관리를 지원한다. 정보저장소는 CASE 프로젝트 데이타베이스이지만 단순히 관련된 정보의 집합이 아니기 때문에 데이타베이스라는 말대신 정보저장소라고 불린다.

2.1 정보저장소 개념

정보저장소는 소프트웨어 생명 주기 동안 모아진 모든 시스템 정보가 관리되고 저장되는 곳으로서, 각 도구들, 각 개발 단계들, 사용자들, 응용 프로그램들 사이의 시스템 정보를 공유할 수 있도록 해준다. 또한 정보저장소에 모든 시스템 요소들과 시스템 정보가 저장되기 때문에 소프트웨어 시스템 유지 보수를 용이하게 해 준다. 그리고 정보저장소는 재사용 가능한 소프트웨어 요소들의 목록을, 이들을 쉽게 이해하고 접근할 수 있는 방법과 함께 유지함으로써 소프트웨어 재사용을 가능하게 해 준다. 정보저장소의 이러한 장점들은 소프트웨어 비용을 줄이는데 매우 큰 역할을 한다. 정보저장소에 저장되는 정보의 종류를 정리하면 다음과 같다.

- 논리적 자료 및 처리 모형
- 물리적 정의와 구현 코드
- 기업(Enterprise) 모형
- 프로젝트 자료와 규칙
- 개체간의 관계
- 정보저장소 메타모델과 검증 규칙

－프로젝트 관리와 감사 정보

정보저장소는 단지 소프트웨어 개발만이 아니라, 모든 소프트웨어 활동을 지원한다. 응용 프로그램 개발, 유지 보수, 시스템 관리 전반에 걸친 모든 활동들을 효율적으로 해준다. 사실상 정보저장소는 소프트웨어 시스템을 만들고 수정하고 강화하고 수행하고 유지하는 데 필요한 모든 정보를 저장하는 곳이라고 할 수 있다.

현재의 정보저장소의 형태는 기업 규모의 정보저장소(Enterprise Repository) 형태로 개발되고 있다. 대부분의 기업들은 기업과 관련된 모든 정보를 매우 가치있는 정보로 간주하기 때문에 이들 정보를 보호, 관리하고 이들 정보를 생산하는 시스템을 유지하는 기법이 필요한데 이 기법이 바로 정보저장소이다. 즉, 정보저장소는 기업 정보의 통합, 기업 정보의 보호, 이들 정보를 만들어 내는 시스템의 관리에 책임을 지고 있다.

기업 규모의 정보저장소는 자료 사전 통합기, 소프트웨어 도구 통합기, 소프트웨어 시스템 통합기, 소프트웨어 생명 주기 통합기로 볼 수 있다.

－자료 사전 통합기

: 라이브러리나 자료 사전과 같은 여러가지 자료 저장소에 정보를 저장하게 되면 자료의 일관성을 유지하기가 어렵고 최근의 자료로의 수정이 쉽지 않은데, 모든 정보를 하나의 정보저장소에 저장함으로써 정보의 정의, 관리를 한 지점에서 할 수 있게 되고, 또한 자료의 일관성과 무결성을 유지시켜 준다.

－소프트웨어 도구 통합기

: 각 소프트웨어 생명 주기의 각 단계의 활동을 지원하는 CASE 도구를 포함한 모든 소프트웨어 도구들을 통합하여 줌으로써, 각 도구들 사이에 정보를 공유할 수 있게 해준다.

－소프트웨어 시스템 통합기

: 대부분의 기업의 목적은 통합된 응용 시스템의 지원을 받는 것이다. 이러한 응용 시스템의 통합은 정보가 저장되고 공유되는 방식을 정의하는 데이타 모델위에서 가능하게 된다. 이러한 기업 규모의 데이타 모델은 전 기업을 통해서 사용되는 모든 시스템을 결합시켜 주는 내부 구조가 되며, 이러한 데이타 모델이 정보저장소에 저장된다.

－소프트웨어 생명 주기 통합기

: 소프트웨어 생명 주기의 각 단계를 이어줌으로써 소프트웨어 시스템을 개발하고 유지하는 각 단계를 통합시켜준다. 소프트웨어 개발의 각 단계를 정의하는 큰 틀을 소프트웨어 개발 방법론이라고 하는데, 정보저장소는 이러한 방법론에 대한 정보를 저장함으로써 사용자에게 그 방법론의 정확한 사용을 가능하게 해줌과 동시에 그 방법론의 각 단계를 지원하는 가장 적당한 도구들의 선택도 가능하게 해준다.

2.2 CASE 도구 통합

CASE 기술에 있어서 가장 중요한 요구 사항이 바로 도구 통합이다. 도구 통합은 소프트웨어 개발 도구들을 효율적으로 연결시켜 주는 중요한 기법이다. 도구 사용자의 관점에서 볼 때 통합 환경내에서 각 도구들은 서로 협력하는 것처럼 보인다. 이렇게 통합된 도구들은 소프트웨어 생산성과 질을 높여주며, 소프트웨어 개발 주기의 다음 단계로 필요한 형태의 정보를 자동적으로 전달하여 줌으로써 소프트웨어 생명 주기를 연결시켜 준다.

통합된 CASE 도구 환경은 소프트웨어 생명 주기 전체를 연결시켜 주는 각 도구들의 집합체로서 다음과 같은 성질을 가지고 있다.

- 공통된 사용자 인터페이스를 가지고 있다.
- 공통된 정보저장소를 사용한다.
- 메타 모델을 가지고 있다.
- 소프트웨어 생명 주기 모델을 지원한다.

공통된 사용자 인터페이스는 통합된 소프트웨어 환경에서의 각 CASE 도구들을 연결하여 주는 다리 역할을 한다. 이는 사용자에게 일관된 느낌을 줌으로써 사용자들이 각 도구를 보다 쉽고 빨리 배울 수 있게 해 준다. 그래픽 사용자 인터페이스에 대한 여러가지 표준이 출현하게 되었는데, OSI의 Motif, X-11의 Windows, IBM SAA의 Presentation Manager, DEC의 DEC wi-

ndows 등이 그것이다.

자료를 공유하기 위해서는 각 도구들이 다른 도구로부터 받은 자료를 인식하고 이해하여야 한다. 각 도구들이 자료 형태와 의미를 잘 이해할 수 있도록 공통된 틀을 제공하는 것이 정보저장소 메타 모델이다. 통합된 도구들은 하나의 메타 모델에 의하여 지원된다.

2.3 정보저장소 메타 모델

정보저장소는 기업, 소프트웨어 생명 주기 과정, 응용 시스템에 관한 모든 정보를 포함한다. 정보저장소에 저장될 정보의 종류, 이들 정보의 형태와 성질, 이들 정보에 접근하고 정확성을 증명하는 방법 등을 정의하는 것이 메타 모델이다. 메타 모델은 정보저장소 전체 내용의 개념적인 관점이다.

메타 모델은 도구 통합과 정보저장소에 저장되어 있는 정보의 무결성의 기초가 된다. 이는 각 도구들이 공통된 자료와 이 자료를 운용하는 방법을 공유할 수 있도록 해 준다. 도구들이 정보를 공유하기 위해서는 같은 형태의 객체 또는 관계와 같은 정보를 상호 호환적인 방법으로 정의하여야 한다. 예를 들어, 한 도구가 자료 흐름도에 대한 정보를 다른 도구에게 넘겨주기를 원한다면, 이 두 도구들은 그 자료 흐름도를 구성하는 각 요소들을 표시하고 이해하는 데 똑같은 방식을 사용하여야 한다.

메타 모델은 정보저장소에 기초를 둔 CASE 도구들이 어떤 소프트웨어 생명 주기와 방법론을 지원하는 지를 결정한다. 소프트웨어 개발 방법론은 어떤 정보가 어떤 형태로 언제 생산되고 사용되는지를 결정한다. 정보저장소가 그 방법론을 지원하기 위해서는 그 방법론이 생산해 내고 사용하는 형태의 정보를 포함하여야 한다. 또한, 메타 모델에는 그 모델의 무결성을 보장할 수 있도록 정보저장소 내용의 사용을 제어할 수 있는 규칙과 제한 조건이 포함되어, 정보저장소의 내용이 들어오고 다루어질 때마다 각 규칙과 제한 조건이 자동적으로 그 내용에 적용된다.

메타 모델이 표현되는 형태는 크게 두가지로 나눌 수 있는데, 개체-관계 모델(Entity Relationship Model)과 객체 지향 모델(Object-Oriented Model)이 그것이다.

개체-관계 모델에서 개체(Entity)란, 사람, 사업 과정, 자료 요소와 같은 것들의 추상화된 표현이다. 관계(Relationship)란, 한 고용인이 한 부서에 속한다 또는 한 프로그램이 부프로그램을 부른다와 같은 두 개체 간의 직접적인 관계이다. 이 개체와 관계를 이용하여 정보저장소의 내용을 나타내게 되는데, 이러한 방법이 정보저장소 메타 모델을 나타내는 데 널리 사용되는 이유는 많은 소프트웨어 응용 시스템을 모델링하는데 이 개체-관계 모델이 자주 사용되기 때문이다. 응용 시스템과 같은 방식으로 정보저장소 내용을 모델링 함으로써 소프트웨어 시스템과 정보저장소의 내용을 보다 빨리 이해할 수 있게 해준다.

객체 지향 모델은 객체(Object) 자료와 이 자료에 작동하는 방법(Method)을 기초로 모델링하는 방법으로서, 정보저장소의 내용을 객체들의 계층 구조로 나타낸다. 소스 코드나 사업 과정 등이 객체의 예가 될 수 있으며, 편집, 복사, 삭제와 같은 자료 관리 연산자는 방법의 예이다.

객체는 제어와 확장성, 자료 관리, 정보저장소 내용의 다른 관점 제공, 복잡한 자료를 다루는 데 있어서의 기초가 된다. 개체-관계 모델이 낮은 수준의 각각의 정보를 정의할 수 있게 해 주는 반면, 객체 지향 모델은 여러 가지 정보와 이에 관련된 연산을 한 곳으로 모을 수 있게 해 준다. 도구들은 수행될 대상이 되는 객체와 방법을 포함하는 메시지를 정보저장소에 보냄으로써 객체 자료를 다룬다.

객체 지향 모델은 변화된 요구 사항에 대처하기 위하여 객체들을 수정하고 새로운 객체들을 더하기 쉬우며, 또한 새로운 도구를 첨가하기가 용이하다. 그리고, 객체 지향 모델은 똑같은 일을 하는 방법의 공유를 가능하게 해준다.

정보저장소는 개체-관계 모델과 객체 지향 모델 두 가지를 다 제공하기도 한다. 여기에서 객체 지향 모델은 프로그램 소스 코드, 목적 코드, 문서와 같은 고수준의 객체에 대한 관점을 보여주고, 개체-관계 모델은 도구와 방법에 의해서 직접 처리되는 낮고 자세한 수준의 개체들의 관점을 제공한다.

메타 모델은 각 도구들과 방법론에 의해서 사용되고 저장되는 정보의 형태와 의미를 제한하는 역할을 하고 있다. 메타 모델은 새로운 도구, 기술, 방법론의 도입으로 인한 새로운 정보의 추가를 지원해야 한다. 따라서, 메타 모델은 사용자가 새로운 객체 형태, 방법 등을 기존의 메타 모델에 더할 수 있도록 확장 가능해야 한다.

정보저장소는 대체로 ANSI SPARC의 3가지 계층의 구조에 기초를 두어 구현된다. 첫째는 논리적 모델로 각 개인 사용자나 각 도구들에 특별한 관점을 나타낸다. 두번째 계층은 개념적인 모델로, 실제적인 구현과는 독립적으로 정보저장소의 전체 내용을 표현한다. 세번째 계층은, 자료 저장의 실제적인 구현을 표현하는 물리적인 모델이다. 이 세가지 계층을 정리하면 다음과 같다.

- ANSI SPARC의 3단계 스키마
  - 논리적 모델 : 특정 사용자(도구) 관점에서의 자료모형
  - 개념 모델 : 전체 정보저장소의 모형
  - 물리적 모델 : 물리적으로 저장되는 자료의 표현

이렇게 계층화된 정보저장소의 구조는 각각의 사용자들에게 다른 관점의 정보저장소 내용을 볼 수 있도록 해 준다. 정보저장소의 3가지 계층 중에서 메타 모델은 개념적인 모델에 해당하는데 보통 개체-관계 모델로 표현된다. 물리적인 계층에서는 정보저장소의 내용이 실제적으로 어떻게 저장되어 있는가를 묘사하며 보통 관계형 모델로 표현된다.

2.4 저장정보의 예

소프트웨어 개발이 진행되면서 소프트웨어 생명 주기 각 단계마다 여러가지 산물들이 생성된다. 그중 어떤 산물은 다음 생명 주기 단계의 입력으로 사용된다. 이 산물들을 정보저장소에서 관리함으로써 자료가 중복되어 저장되는 것을 막고 도구들 간에 자료를 공유하는 것이 용이해진다. 구조적 기법을 따르는 소프트웨어 개발 과정에서 생성되는 산물과 그들 간에 어떤 관계가 있는지, 그것들이 어떻게 정보저장소에 저장될 수 있는가에 대한 개념적인 관점을 제시하겠다.

2.4.1 분석 단계

개발하고자하는 소프트웨어의 기능에 관한 요구 사항 명세서를 작성하는 단계이다. 이 단계에서는 주로 명세 관련 문서를 작성하게 되는데 요구 사항 문서, 자료 처리의 흐름을 나타내는 자료 흐름도, 자료 모델링을 위해서 사용되는 객체 관계도, 자료에 대한 정의를 가지고 있는 자료 사전 등이 생성된다. 분석 단계에서는 문서를 관리할 수 있는 기능을 가진 도구, 자료 흐름도 작성을 지원하는 도구, 자료 흐름도 내 프로세스의 기능을 설명하는 프로세스 명세 작성을 지원하는 도구, 객체 관계도 작성을 지원하는 도구, 자료 사전 작성을 지원하는 도구, 그 외에도 결정표나 상태 전이도를 작성하도록 도와주는 도구가 필요하다.

2.4.2 설계 단계

이 단계에서 주로 생성되는 정보는 설계 문서이다. 구조적 방법론을 사용하여 설계를 할 경우 분석 단계의 산물인 자료 흐름도로부터 구조도를 생성하게 된다. 설계 단계에서는 구조도 작성을 지원하는 도구, 자료 흐름도로 부터 구조도로 변환하는 도구, 구조도의 각 모듈과 관련된 모듈 명세 작성을 지원하는 도구 등이 필요하다.

2.4.3 코딩 단계

코딩 단계의 산물은 원시 코드이다. 원시 코드는 독립적으로 작성될 수도 있고 구조도의 모듈 명세로부터 변환될 수도 있다. 모듈 명세로부터 변환될 경우 완전한 코드가 되기는 어렵고 코드 프레임의 형태를 가지게 된다. 이 단계에서는 원시 코드를 편집하기 위한 편집기, 모듈 명세를 코드 프레임으로 변환하는 도구, 원시 코드를 컴파일하기 위한 컴파일러, 디버깅을 위한 디버거가 필요하다. 원시 코드를 분석할 수 있는 분석기도 이 단계에서 유용하게 사용되는 도구이다. 코딩 단계에서 사용되는 도구들은 대부분 프로그래밍 언어의 문법을 기반으로 기능을

수행한다.

2.4.4 시험 단계

개발된 소프트웨어가 원하는 기능을 올바르게 수행하는지의 여부와 그 성능을 시험해 보는 단계이다. 이 단계에서는 시험 자료와 시험 결과 자료가 생성된다. 시험 단계에서는 시험 자료를 생성하는 도구, 각 프로그래밍 언어를 지원하는 분석기가 있다.

2.4.5 유지 보수 단계

이미 개발된 소프트웨어를 여러가지 이유로 유지 보수할 필요가 생긴다. 이 단계에서는 역공학 관련 도구들에 의해서 원시 코드 분석 정보가 생성된다.

지금까지 구조적 기법에 의한 소프트웨어 개발 과정에서 생성되는 산물들을 간략하게 살펴보았다. 이 정보들을 저장하기 위하여 이들을 위한 메타모델을 정의할 수 있다. 정보저장소 메타모델에서 지원해야 할 정보의 유형은 다음과 같다.

- 프로젝트 (Project)
- 자료 흐름도 (Data flow diagram, DFD)
- 구조도 (Structure chart, SC)
- 객체 관계도 (Entity relationship diagram, ERD)
- 자료 사전 (Data dictionary, DD)
- 프로세스 명세 (Process specification)
- 모듈 명세 (Module specification)
- 결정표 (Decision table)
- 상태 전이도 (State transition diagram, STD)
- 원시 프로그램 (Source program)
- 문서 (Document)
- 주석 (Comment)
- 개발 비용 (Cost)
- 개발 일정 (Schedule)
- 사용자 정보 (User information)
- 기타

정보저장소에는 여러개의 프로젝트가 속할 수 있으며 모든 정보는 프로젝트별로 관리된다. 문서는 사용자가 작성한 프로젝트에 관련된 사항이며 개발 비용과 개발 일정은 프로젝트가 진행중일 경우는 추정치를 가지고 있으며 프로젝트가 완결되었을 경우 추정치와 실제치 두 가지를 가진다. 사용자 정보는 프로젝트에 대한 소유자, 접근 권한 등을 명시한 것이다. 기타 정보에 속하는 것으로서는 프린터 출력을 위하여 그래픽 정보(자료 흐름도, 구조도 등)를 PostScript로 변환한 화일, 각종 정보를 검증한 결과를 담고 있는 화일, 오류 메시지, 임시 화일 등이 있다.

3. 정보저장소의 기능과 구조

시스템 정보의 자동적인 관리는 그것을 컴퓨터에 저장한 주요 목적이며 정보저장소가 제공하는 가장 큰 장점이다. 비형식적이고 수작업인 방법으로 정보를 관리하면 저장된 정보의 무결성을 유지하기가 어렵다. 특히, 큰 프로젝트 팀이나 시스템에서는 더욱 그러하다. 정보를 안전하게 공유하고 또 자동화 방법으로 관리하는 능력이 정보저장소에 필요하다.

논리적으로 정보저장소는 하나의 저장 장소로 간주됨으로써, 정보저장소의 내용을 쉽게 관리하고 제어할 수 있다. 그러나 물리적으로는 하나 또는 여러 개의 저장 장소로 구성된다. 정보저장소의 물리적인 구조는 정보저장소의 판매자나 사용자의 선택에 의해서 중앙집중적인(Centralized) 구조와 분산된(Distributed) 구조로 나누어진다.

3.1 기 능

정보저장소는 저장된 정보를 사용자간에 무결성을 유지하면서 공유할 수 있도록 하기 위하여 정보 관리 기능을 가지고 있다. 정보저장소가 제공하는 정보 관리 기능을 설명하면 다음과 같다.

3.1.1 보안성 관리

정보저장소는 저장되어 있는 내용을 접근하는데 있어서 보안성을 보장하여야 한다. 각 사용자에게 정보를 읽고, 수행하고, 수정하고, 지우는 각각의 작업에 대해서 접근 권리를 부여함으로써 저장되어 있는 정보를 안전하게 보호한다.

3.1.2 버전 관리

정보저장소에는 정보의 여러 개의 복사본이 존재하게 되는데 가능한 한 많은 수의 복사본이 존재할 수 있도록 해 주는 것이 버전 관리이다. 버전 관리의 대상은 하나의 기능을 수행하는 프로그램이나 모듈과 같은 개체로 정할 수 있다.

3.1.3 변경 관리

정보저장소의 변경 관리 기능은 정보저장소의 요소들의 변화를 관리하는 것이다. 정보저장소는 각 요소들 뿐만 아니라 그들의 관계도 저장하므로, 변경 관리 기능은 어떤 변화로 인해 영향을 받는 요소들의 범위에 대한 정보를 제공하여 준다. 변경 관리 기능은 한 요소가 바뀌면, 이 변화에 영향을 받는 모든 요소들을 알아낼 수 있도록 해 주며, 또한 한 요소의 정의가 바뀌면 그 요소에 접근하는 모든 사용자들에게 이들의 변화를 알려 준다. 어떤 경우에는 한 요소가 바뀌면 모든 버전과 이와 관련된 모든 요소들을 자동적으로 바꾸어 준다.

정보저장소의 각 요소들이 변하면 그에 따라서 항상 기록해야 하는 정보가 있는데, 누가, 언제, 왜 요소들을 바꾸었느냐에 관한 정보 등이다. 또한 그 요소의 소유자, 변경의 날짜, 상태에 관한 정보 등도 유지된다.

3.1.4 형상 관리

형상 관리란 소프트웨어 시스템의 모든 요소들을 이해하고 수행시 각 요소들을 어떻게 결합시켜야 하는지에 관한 정보를 유지하는 것이다. 형상 관리는 각 요소들이 최근의 버전을 유지하고 있음을 보장하여야 한다. 또한 하드웨어나 소프트웨어의 실패로 인해 유실할 수 있는 정보저장소의 내용을 보호하기 위하여 백업과 복구 기능을 포함하여야 한다.

3.1.5 질의와 응답 기능

사용자에게 정보저장소의 내용을 볼 수 있게 하기 위해서는 질의 및 응답 기능을 제공해 주어야 한다. 어떤 경우에는 SQL 또는 IRDS질의 언어가 정보저장소의 정보를 추출하는 질의 언어로 사용된다. 여러 종류의 표준화된 응답 형식이 제공되어 정보저장소의 내용을 쉽게 볼 수 있도록 해 준다. 예를 들면, 각 자료가 시스템의 어디에서 사용되는지, 누가 만들었는지, 언제 그리고 얼마나 자주 그 정의가 바뀌었는지에 관한 정보가 이 응답 형식에 제공된다.

3.1.6 다중 사용자 지원

정보저장소는 여러 사용자가 동시에 똑같은 정보에 접근할 수 있도록 해 주어야 하며 여러 사용자에 의한 정보의 수정과 삭제를 제어하여야 한다. 다중 사용자를 지원하는 한 기법으로 한 사용자가 수정하려고 하는 정보저장소 요소를 시스템에 표시하고 가져가면, 다른 사용자는 이 요소에 수정을 가할 수 없도록 하는 방법이 있다.

3.1.7 개방형 구조(Open Architecture)

각 도구들이 정보저장소의 정보 관리 기능을 사용할 수 있도록 인터페이스에 관한 정보를 제공해 주어야 한다. 또한 한 정보저장소로부터 다른 정보저장소나 데이타베이스 또는 자료 사전으로의 정보 교환도 가능하게 해 주어야 한다.

3.1.8 검증

정보저장소의 관리 기능은 저장된 내용의 정확성을 검증할 수 있는 기능을 제공하여야 한다. 정보의 정확성을 검증하는 여러가지 종류의 방법이 정보저장소 메타 모델에 정의되어 있는데 이를 규칙 또는 정책이라고 한다. 이러한 규칙이나 정책은 정보저장소 요소가 처음 정보저장소 안으로 들어오거나 변경될 때 지켜야 하는 제한 조건이며 요구 사항이다. 이러한 규칙이나 정책을 메타 모델에 포함시킴으로써 정보저장소를 사용하는 각 도구들을 단순화시키며 검증 과정을 표준화시킬 수 있다.

3.2 정보저장소의 구조

논리적으로 정보저장소는 하나의 저장 장소로 간주됨으로써, 정보저장소의 내용을 쉽게 관리하고 제어할 수 있다. 그러나 물리적으로는 하나 또는 여러개의 저장 장소로 구성된다. 정보저장소의 물리적인 구조는 정보저장소의 판매자나

사용자의 선택에 의해서 중앙집중적인 구조와 분산된 구조로 나누어 진다.

물리적으로 중앙집중된 구조에서의 정보저장소는 메인프레임 단계에 존재한다. 즉, 모든 정보저장소의 내용이 메인프레임에 존재하고 이들을 이용하는 도구들은 소프트웨어 도구 환경을 이루는 요소의 어느 곳에도 존재할 수 있다. 이렇게 함으로써 중앙의 자료 관리 시스템이 제공하는 자료의 무결성, 복구, 보안성 보장 등의 기능을 잘 이용할 수 있다.

물리적으로 분산된 구조에서의 정보저장소의 내용은 여러 단계로 나누어진 자료 저장 장소에 분산되어 저장된다. 정보저장소의 각 부분들이 물리적으로는 네트워크의 각 장치에 분산되어 있지만, 논리적으로는 하나의 정보저장소로 간주된다. 분산된 구조를 갖는 정보저장소는 보다 효율적이며 신뢰성이 높다. 분산된 구조에서 생산성을 높이기 위해서는 각 소프트웨어 개발 활동, 각 생산 환경, 각 부서마다 분리된 물리적 정보저장소를 두면 된다.

4. 정보저장소 표준화 동향

정보저장소 간의 정보 교환뿐만 아니라 소프트웨어 도구의 통합, 여러 제작자가 만든 도구의 통합을 가능하도록 하기 위하여 정보저장소 표준화는 중요하다. 본 절에서는 다음의 정보저장소 표준에 대해 간략하게 설명할 것이다. 자세한 내용은 참고문헌을 참조하기 바란다.

- CASE Data Interchange Format (CDIF)
- Information Resource Dictionary System (IRDS)
- IBM SAA de facto Standard
- A Tools Integration Standard (ATIS)
- Portable Common Tool Environment (PCTE)

4.1 CASE Data Interchange Format (CDIF)[3]

CDIF는 CASE 도구들 간의 정보 공유를 가능하게 하기 위한 언어에 대한 명세이다. 이것이 정확히 정보저장소 표준은 아니라고 하더라도, CDIF는 다른 곳에 있는 정보를 정보저장소로 가져오거나, 정보저장소의 정보를 CASE 도구에 보낼 때 유용할 것이다. CDIF는 CAD/CAM/CAE 도구들 간의 정보 공유를 위해 개발된 EDIF 표준의 확장이며, EDIF/CASE로 불리기도 한다. CDIF가 도구들 간의 정보 공유를 위해 사용하는 수단은 CASE 도구에서 출력되거나 CASE 도구로 입력되는 ASCII 문자 화일이다. 이 화일 자체는 정보에 대한 설명뿐만 아니라 시스템 정보도 포함하고 있으며, 문자뿐만 아니라 그래픽 정보도 가지고 있다.

현재 200명의 사용자와 CASE 도구 제작자, 컴퓨터 제작자, 정부 관리, 학술 기관이 CDIF 위원회에 활동하고 있고, 약 30명이 능동적으로 참여하고 있다. EDIF는 CDIF를 공식적으로 인정했으며 CDIF로 발전하고 있다.

4.2 Information Resource Dictionary System (IRDS)[4,5]

IRDS는 National Institute of Standards and Technology에서 개발한 정보저장소 표준이다. 이것은 정보저장소의 내용, 사용자 인터페이스와 정보저장소 간에 정보를 옮기기 위한 인터페이스 등을 정의하고 있다. IRDS를 이해하는 데 가장 좋은 방법은 비지니스 수준의 설계로 보는 것이다. IRDS는 모든 정보저장소의 내용과 입력, 출력을 표현하고 있지만, 기술적인 설계나 구현에 대한 계획은 표현하고 있지 않다. 그것은 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 본다.

IRDS 표준은 다음의 여섯 모듈을 포함한다.

- Core Dictionary System
- Basic Functional Schema
- IRDS Security
- Extensible Life Cycle Phase Facility

- IRDS Procedure Facility
- Application Program Interface

4.3 IBM SAA 와 AD/Cycle[6]

AD/Cycle은 IBM이 개발한 응용 시스템 개발 환경이다. 이것은 생명 주기의 각 단계를 지원하는 소프트웨어 도구들과 서비스들로 이루어져 있으며, 여기에는 비지니스 모델링(Business Modeling), 분석, 설계, 구현, 검사 및 유지 보수가 포함된다. AD/Cycle의 도구들과 이것을 이용해서 개발된 응용 시스템들은 SAA(Systems Application Architecture) 지침을 따른다.

SAA는 통신(Common Communications Support, CCS), 사용자의 접근(Common User Interface, CUI), 응용 시스템의 개발(Common Program Interface, CPI와 AD/Cycle)을 포함하는 객체들과 표준들의 모임이다. SAA 시스템은, MVS, VM을 사용하는 IBM 대형 컴퓨터, OS/400을 사용하는 AS/400, OS/2를 사용하는 PS/2 등의 시스템들을 포함한다. AD/Cycle은 SAA 환경 전체에 걸쳐서 포괄적이고 통합된 응용 시스템 개발의 틀을 제공하는 기반이 된다. 이러한 SAA 환경은 다음을 제공하고 있다.

- 생명 주기 전체를 지원
- 공통된 사용자 인터페이스
- 공통된 정보저장소
- 개방형 구조

AD/Cycle은 응용 시스템 개발과 유지 보수를 개선하기 위한 하나의 SAA 방법이며, 정보저장소에 기반을 둔 환경이다. IBM Repository Manager와 그에 연관된 AD Information Model은 모든 응용 분야에 관련된 정보의 공통적인 표현 방식을 제공한다.

AD/Cycle의 도구들은 IBM과 그 협력업체들(예를 들면, Intersolv, KnowledgeWare, Bachman, Systematica, Synon)에 의해 제공된다. 어떤 도구들은 생명 주기의 한 단계만을 주로 지원하고(예를 들면, 모델링 도구, 화면과 보고서 작성기, 코드 생성기, 검사의 범위 분석기 등), 어떤 도구들은 생명 주기 전체에 사용된다. 생명 주기 전체에 적용되는 도구들의 예로는 프로젝트와 처리 관리 도구, 영향 분석 도구와 도큐멘트 도구가 있다.

AD/Cycle은 개방형 구조로 된 환경이므로, 고객과 도구의 제작자는 AD/Cycle의 환경에 통합할 도구들을 제작하고 통합할 수 있다. 이런 AD/Cycle의 개방적 구조는 도구들이 사용할 인터페이스와 보편적인 서비스에 대해 공개적으로 정의하고 있다. AD Information Model은 상업적으로 구입할 수 있으며, 도구와 사용자의 새로운 정보에 대한 요구를 수용할 수 있도록 확장이 가능하다.

4.4 A Tools Integration Standard (ATIS)[2]

ATIS는 Integrated Project Support Environment (IPSE)를 위한 객체 지향적 정보저장소 인터페이스이다. 이것은 현재 Atherton Technology의 Software BackPlane으로 불리는 제품에서 이용되고 있다. Digital Equipment Corporation에서는 ATIS를 그들의 정보저장소인 CDD/Repositoy에 합병시켰고, ATIS를 IRDS 정보저장소 표준에 합병시켜야 한다고 제안하였다. IBM 또한 ATIS를 그들의 AIX(UNIX) CASE 전략에 포함시킬 계획을 갖고 있다.

ATIS 표준은 정보저장소의 인터페이스를 정의하고 있는데, 이렇게 함으로써 이 인터페이스를 사용하는 정보저장소들은 상호 통신할 수 있다.

ATIS 표준은 과학과 공학의 시장을 목표로 하고 있으며, IBM의 SAA처럼 전체적인 통합의 틀을 제공하고 있다. 원래 이 계획은 Software BackPlan에서 전체 틀을 위해 개발되었지만, 그 이후로 접근 방식이 바뀌어 여러 가지의 통합 방식을 제공하게 되었다. 이렇게 고려되고 있는 통합 방식들 중에는, 버젼 관리, 형상 관리, 도구들 간의 통신, 소프트웨어 객체들 간의 관계, 도구의 추가를 위한 동적인 확장성 등이 있다. BackPlan에 대해 알아 보면, 데스크 탑(Desktop) 방식과 객체 지향적 접근 방식을 사용하였으며, 지원되는 기능에는, 목록 지원(Catalog Support), 제어 통합(Control Integration), 일의 흐름 제어, 자료 통합(Data Integration), 사용자-모델 통합

(User-Model Integration), 그리고 도구 통합을 위한 점진적인 처리 등이 포함되어 있다. 통합에 있어서 이러한 일을 추구하고 있는 평의회의 이름은 CIS(CASE Integration Services)이다.

4.5 Portable Common Tool Environment (PCTE)[10]

PCTE는 소형 컴퓨터/LAN 환경에서 CASE 도구의 이식성을 높이기 위한 목적을 가진 인터페이스 표준이다. 이들의 생각은, 운영 체제와 CASE 도구 사이에서 수행되는 함수들의 집합을 명시함으로써, CASE 도구가 운영 체제를 호출하기 보다는 이 함수들을 호출하도록 하는 것이다.

각각의 운영 체제의 환경에 대하여, CASE 도구는 변하지 않은 채, 단지 이 PCTE 함수의 집합만 변화한다. CASE 도구는 PCTE 인터페이스를 호출하고, 이 PCTE 인터페이스는 다시 운영 체제를 호출한다. 만약 같은 CASE 도구가 다른 운영 체제 하에서 수행되고 있다 하더라도, 그것은 똑같은 PCTE 호출을 하고, PCTE 인터페이스는 그 호출을 운영 체제 명령에 알맞게 번역한다. 이런 방식으로 수행 환경 사이의 이식성이 달성된다.

PCTE가 소형 컴퓨터 LAN 환경을 위해 만들어졌고, 큰 규모의 개발을 지원하고 있으므로, PCTE는 팀 개발을 지원할 수 있는 정보저장소의 개념을 가지고 있다. PCTE는 운영 체제에 따라서 각각의 워크스테이션에 이식된다. PCTE의 특성은 다음과 같다.

- 문자와 고해상도의 그래픽을 모두 포함하는 다중 윈도우 관리 기능
- 연결되어 있는 어떤 워크스테이션에서라도 명령의 수행이 가능하도록 하는 분산 처리
- 분산 저장과 연결된 워크스테이션으로부터의 정보의 추출

PCTE는 프로젝트 수준의 정보저장소의 역할을 할 수 있다. PCTE+는 국방 응용 분야를 지원하기 위한 확장이고, 따라서 접근에 대한 제어와 다른 보호 기능들이 있다.

5. 결 론

CASE 환경을 구축하여 소프트웨어 생산성과 품질을 향상시킬 수 있다. CASE 환경은 여러종류의 도구들이 포함되어 프로젝트를 지원한다. 이 도구들이 생성하거나 사용하는 정보를 저장하고 관리할 필요가 있다. 이 정보들을 공통적으로 표현하고 관리하기 위한 정보저장소를 구축하여 도구 상호간에 정보를 공유할 수 있도록 하는 것이 중요하다. 여러 도구들이 다루는 정보의 종류는 메타모델로써 서술할 수 있으며, 여기에는 정보에 관한 규칙이 함께 정해져서 무결성을 유지한다. 정보저장소는 정보 관리 기능과 함께 질의 기능도 가져야 한다. 지금까지 정보저장소의 기능과 구조, 그리고 표준화 동향에 관하여 기술하였다. 독립적으로 개발되는 정보저장소가 공통되는 양식을 가진다면 상호간에 정보를 교환할 수 있다.

참고문헌

[1] 우치수 외, "CASE Tool 개발 연구", 상공부, 1992.

[2] "A Tools Integration Standard," Digital Equipment Corporation, Feb, 1990.

[3] Alan Hecht and Matt Harris, "A CASE Standard Interchage Format: Proposed Extension to EDIF 200," Cadre Technologies, Inc., White Paper.

[4] ANSI, "A Technical Overview of the IRDS system, Annex 1B, Draft," ANSI, 1985.

[5] ATIS, ANSI X3H4 Working Draft, Information Resource Dictionary System, Feb, 1990.

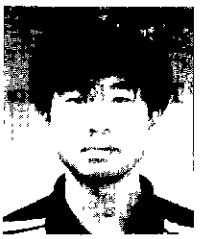
[6] Henry C. Lefkovits, IBM Repository Manager/MVS, Q.E.D. Technical Publishing Group, 1991.

[7] Ian Thomas, "Tool Integration in the Pact Environment," proc. of the 11th International Conference on Software Engineering, pp. 16~18, May, 1989.

[8] Leon J. Osterweil, "Toolpack-An Experimental Software Development Environment Research Project," IEEE Trans. on Software Enginee-

ring, pp. 673~685, Nov, 1983.
[9] N. Meyerowitz, "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework," OOPSLA '86 proc., pp. 186~201, 1986.
[10] PCTE, PCTE Functional Specifications 1.4, Bull and *et.al*, Sep., 1986.
[11] S. Reiss, "Connecting Tools Using Message Passing in the Field Environment," IEEE Software, pp. 57~66, Jul., 1990

강 병 도



1986 서울대학교 계산통계학과 졸업(전산과학 전공)
1988 서울대학교 대학원 계산통계학과 졸업(전산과학 전공, 이학석사)
1993 서울대학교 대학원 계산통계학과 박사과정 수료(전산과학 전공)
1988 ~현재 한국전자통신연구소 연구원
관심 분야 : 소프트웨어 개발 방법론, 개발환경 및 CASE 환경 구축

우 치 수

1972 서울대학교 공과대학 졸업
1977 서울대학교 대학원 이학석사(전산과학 전공)
1982 서울대학교 대학원 이학박사(전산과학 전공)
1972 ~1974 KIST 연구원
1978 영국 라퍼러대학 연구원
1974 ~1982 울산공대 전산학과 조교수, 부교수
1985 미국 미시간 대학 Post-doc
1982 ~현재 서울대학교 계산통계학과 조교수, 교수
관심 분야 : 소프트웨어 공학, 프로그래밍 언어