소프트웨어 통합환경구축을 위한 도구의 분석과 설계 방안

양 해 술† · 이 하 용††

요 약

소프트웨어의 통합환경을 구축하기 위해 통합 프로젝토 지원환경(IPSE: Integrated Project Support Environment)을 구축할 때에는 2가지의 문제가 발생한다. 첫째는 도구의 밀도에 관한 문제이다. 여러 가능을 하나로 통합하는 조립 과정을 거친 도구들의 인터페이스에 관한 문제와 도구의 일부 기능을 다른 도구와 함께 사용하는 경우 닫혀진 인터페이스로 인해 이용이 곤란해지는 경우가 발생할 수 있다는 점이다. 둘째는 프로세스에 관한 자료를 수집할 때 발생할 수 있는 문제이다. 개발 프로세스를 향상시키기 위해 이를 측정하려고 할 때 도구를 사용하여 자동적으로 자료를 취득할 수도 있지만 기존의 도구들이 개발 프로세스에 대한 측정을 사전에 배려하여 개발된 것이 아니기 때문에 측정이 곤란하다는 점이다. 본 연구에서는 이와 같은 문제를 해결하기 위해 도구를 정밀도화하고 프로세스 정보를 취득하기 위한 기구를 추가할 필요성을 기술하고 문제해결을 위한 분석과 설계 방안을 모색하였다. 그리고 도구의 결합도와 응집도 및 통합적합성에 대한 메트릭스를 제안하였다. 끝으로 본 연구결과에 기초하여 환경을 구축하는 경우에 일어날 수 있는 문제점에 대한 고찰과 평가를 하였다.

Analysis and Design Method of Tool for Construction of Integrated Environment of Software

Hae-Sool Yang† · Ha-Yong Lee††

ABSTRACT

Two problems can be happened when we construct Integrated Project Support Environment for construction of software integration environment. The first is a problem about granularity of tool. Problems about interface of tools after combination phase which combine many functions into one and when we use some function of tool with other tools, we can't be use by closed interface. The second is a problem that can be happen when we gather data about process. When we measure the development process for enhancement of it, we can get data with tools, but current tools are not prepared to measurement for development process. In this study, we explained that process information tool is needed and proposed analysis and design methodology and evaluation standard. Finally, we considered and evaluated for problems which can be happened when we construct environment which is based on this result of study.

1. 서 론

고품질의 소프트웨어를 한정된 기간내에 계획된 비용 범위내에서 개발하고자 하는 노력은 소프트웨어 공학의 연구 분야에서 중요한 과제이다. 이러한 노력의 일환으로 소프트웨어 개발과정을 자동화함으로써 가능한한 개발 비용을 절감하고 사용자의 요구에 신속히 대처할 수 있는 노력이 지속적으로 연구되어 왔다.

† 종신회원: 한국소프트웨어품질연구소 소장
†† 종신회원: 강원대학교 전자계산학과 박사과정
논문접수: 1996년 2월 9일, 심사완료: 1996년 3월 26일

소프트웨어를 개발함에 있어서 인간의 창의력과 기술적 노하우에만 의존하던 시대를 벗어나 1980년대에 이르러 CASE 도구가 개발됨으로서 소프트웨어의 개발을 부분적으로 자동화할 수 있게 되었다. 그러나 대부분의 CASE 도구들은 생명주기의 일부분만을 지원하고 있는 것이 현실이며 넓은 범위를 지원하는 CASE 도구라 하더라도 도구를 구성하는 모든 요소들이 사용자에게 만족할 만한 결과를 주지 못하고 있다. 따라서 사용자들은 만족할 만한 개발환경을 구축하기 위해서 자신의 요구에 적합한 도구들을 조합하여 필요한 성능을 갖추려는 시도를 하게 되었다. 즉, 사용자가 필요로 하는 다양한 서로 다른 제품의 우수한 기능 요소들을 통합하여 좀더 효율적인 개발환경을 구축하는 것이 중요한 과제라고 할 수 있다. 이러한 통합 프로젝트 지원환경(IPSE : Integrated Project Support Environment)을 구상할 때에 여러 가지 문제점이 발생할 수 있는데 대표적인 것으로 2가지 문제점을 들 수 있다[1].

첫번째는 도구의 밀도(granularity)에 관한 문제이다. 사용자는 만족할 만한 성능을 위해 다양한 기능을 갖는 도구들을 통합하게 되는데 여러 기능을 하나로 통합하는 조립 과정을 거친 도구는 일반적으로 다른 도구와의 인터페이스가 가장 큰 문제로 대두될 수 있다. 조립 과정을 거친 도구를 구성하는 단위 도구는 단위 도구들간의 조립을 위한 인터페이스에 맞추어져 있다. 마찬가지로 조립 과정을 거친 또 다른 도구의 단위 도구는 자신들만의 인터페이스에 맞추어져 있으므로 조립 과정을 거친 도구들간의 통합을 위한 인터페이스의 설정이 어렵게 된다. 즉, 어느 한쪽에 맞게 인터페이스를 수정하려 한다면 조립된 도구의 내부적인 인터페이스까지 수정해야 하기 때문이다. 그러므로 도구가 가진 기능들 중의 일부를 다른 도구와 함께 사용하는 경우를 고려해보면 필요한 인터페이스는 닫혀져 있고 이용이 곤란할 수도 있다. 또한, 도구가 이를 사용하는 프로세스에 대해 최적의 기능을 가지고 있지 않을 수도 있고 어느 정도 제약이 따를 수도 있으므로 사용자가 이 제약에 따를 수 밖에 없는 상황이 발생할 수 있다.

두번째는 프로세스에 관한 자료를 수집하고자 할 때 발생할 수 있는 문제이다. 개발 프로세스를 향상시키기 위해서는 개발 프로세스의 현재 수준 및 문제점을 발견하기 위해 개발 프로세스에 대한 측정이 필요하다. 프로세스를 측정하고자 할 때 가능한한 도구를 사용하여 자동적으로 자료를 취득할 수 있도록 하는 것이 바람직하지만 기존의 도구들은 대부분 개발 프로세스의 자동적인 측정에 대해 사전에 배려하고 있지 않다는 문제점이 있다.

본 연구에서는 이와 같은 문제점을 해결하기 위해 도구의 정밀도화와 프로세스 정보취득기구의 추가적인 필요성을 기술하고, 문제 해결을 위한 분석과 설계 방법 및 평가 기준을 제안하였다. 즉, 제2장에서는 도구 통합시의 문제점과 그 해결 방안에 대해 고찰하였으며, 3장에서는 구체적인 도구의 분석과 설계 방법을 제안하였으며 4장에서는 도구 및 IPSE의 평가 기준에 대해 고찰하였다. 끝으로 5장에서는 지금까지의 연구결과에 기초해서 환경을 구축하는 경우의 새로운 문제점과 그 해결 방안에 대해 기술하였다.

2. 도구 통합시의 문제점과 해결 방안

2.1 도구 통합시의 문제점

적절한 IPSE용의 도구를 만들 때에 SPICE(Software Process Improvement and Capability dEtermination) 모델에 보인 것과 같은 서비스(인터페이스)에 준하는 것 만으로는 불충분하다[1]. 그 이유는 도구 개발자와 제공하는 도구를 이용하여 환경을 구축하는 사용자와는 도구에 대한 입장이 다르기 때문이다. 도구 개발자가 도구를 범용적인 시각으로 보는 반면, 사용자는 자신의 특정한 환경에 적용할 특정한 도구로서 파악하기 때문이다. SPICE는 다음과 같은 6단계의 능력 능력 레벨로 분류하고 있다.

① Level 0 : 아무것도 하지 않는 레벨
② Level 1 : 비형식적으로 실행되고 있는 레벨
③ Level 2 : 계획하여 실행하고 있는 레벨
④ Level 3 : 적절하게 정의되어 있는 레벨
⑤ Level 4 : 정량적으로 관리되고 있는 레벨
⑥ Level 5 : 지속적으로 개선하고 있는 레벨

2.1.1 도구의 조합

동일한 개발자가 개발하지 않은 개개의 도구들은 통합을 위한 각각의 인터페이스에 대해 미리 고려하여 개발되지 않고 다른 전제 상에서 만들어진다. 도

구가 복잡한 조립 과정을 거친 경우 일반적으로 도구에 포함된 각 기능은 특정한 실행순서를 가진다. 이때 도구내에 정의되어 있는 기능의 실행순서 즉, 프로세스의 과정이 사용자 프로세스의 과정과 적합하지 않은 경우가 많다. 특히, 이 부적합성은 소프트웨어 개발 초기에 나타나게 된다. 왜냐하면 상위 단계에서는 하위 단계에 비해 그 절차가 다양하기 때문이다. 만약, 모든 도구 개발자와 환경 구축자 사이에서 특정한 환경에 관한 도구의 기능 및 프로세스에 관해 합의할 수 있다면 필요한 IPSE를 구축할 수 있다. 그러나 도구는 범용으로 만들어지는 반면 사용자 환경은 특정하고 다양하다는 문제점이 있다.

다른 전제에서 만들어진 도구를 동일한 환경에서 동작시키기 위한 유효한 수단은 도구를 정확하게 분석하여 분류하는 것이다. 이와같이 분류된 도구를 각 프로세스와 대응시킴으로써 사용자 프로세스를 대응하는 도구와 조합시키는 방법으로 정의할 수 있다.

2.1.2 프로세스 자료의 취득

SPICE 모델[1]에서는 프로세스 관리 서비스로서 6개의 서비스가 요구되고 있다. 또, CMM(Capabi- lity Maturity Model)[3]에서 레벨은 반복가능(레벨 2), 정의되고 있음(레벨 3), 관리되고 있음(레벨 4), 최적화되고 있음(레벨 5) 이다. 이들 서비스 및 각 레벨의 기본이 되고 있는 것은 정의와 측정이며 정의되고 측정됨으로써 프로세스는 개선된다. 측정할 수 없는 것은 명확하게 정의할 수 없다. 프로세스의 정의에 대해서는 많은 모델화 방법이 제안되어 있지만[4], 여기에서는 그들 정의방법(프로세스 모델화 기법)에 따르지 않고 측정하는 방법에 대해 고려하기로 한다.

종합환경상에서 대부분의 프로세스 측정은 도구를 개입시켜 자동화할 수 있으며 정확한 측정을 위해서도 자동화할 필요가 있다. 대개의 경우 프로세스를 파악해야 하는 중요한 프로젝트 과정은 수동으로 측정 정보를 전달할 시간적인 여유가 없다.

자동적으로 측정할 수 있는 2가지 방법은 도구의 상태로부터 파악하는 방법과 도구와 관계된 객체의 상태로부터 파악하는 방법이며 이와같은 2가지 관점에서 접근하기로 한다.

(1) 도구의 상태를 이용한 측정

어떤 프로세스를 지원하고 있는 도구가 동작 중일 때는 그 프로세스를 활동상태에 있다고 정의하는 방법이다. 이 방법은 기존 도구의 변경이 불필요하고 구현이 용이하지만 정확성과 정밀도가 불충분하다. 정확성이란 도구의 실행과 프로세스의 실행이 일치하는 정도를 말하며 정밀도는 프로세스의 진행상태를 얼마나 상세히 알 수 있는가를 나타내는 것이다. 예를 들면 구조화 분석 도구에서 자료사전을 기술하고 있는 경우, 단지 '분석을 하고 있다'라는 정보밖에 얻을 수 없는 경우에는 정밀도가 낮다고 볼 수 있다.

(2) 객체의 상태와 특성을 이용한 측정

객체의 상태와 특성을 이용한 측정은 도구상태를 이용한 측정에 비해 보다 정확하고 정밀한 프로세스 정보를 얻을 수 있다.

(a) 객체의 상태

객체의 생성/소멸/수정 등에 관한 정보로서 이 정보를 통해 관련된 프로세스가 얼마나 활발히 진행되고 있는가를 추정할 수 있다. 예를 들면, 구조화분석 지원도구에서 각 도형 요소가 객체에 대응하고 있는 경우 설계가 얼마나 진행되고 있는가를 생성되는 객체의 수로 파악할 수 있다. 이와같은 방법은 도구 상태를 이용하는 방법에 비해 보다 정확하고 정밀도가 높다.

(b) 객체의 특성

객체의 특성은 보다 정밀한 프로세스 정보를 줄 수 있다. 특성이라는 것은 예를 들면 프로그램 코드의 행수 등의 메트릭스 정보로 표현되는 것으로서 이것은 객체가 만들어진다는 정보보다도 그 내용을 고려하기 때문에 보다 정밀한 프로세스 정보로 이용할 수 있다. 단, 그들의 값이 사전에 주어져 있지 않은 경우 모든 객체에 대해 그들을 계산할 필요가 있으므로 이 프로세스 정보의 취득에 관련된 비용이 높다는 단점이 있다.

2.2 통합에 관한 문제점의 해결 방안

여기에서는 제2절에서 열거한 2가지 문제점을 해결하기 위한 방법을 검토하기 위해 우선 다음과 같은 점을 고려한다. 통합된 도구는 위에서 기술한 점에

관해 도구 개발자 사이의 합의가 필요하게 되는데 합의를 얻기 위해서는 도구에 관한 참조 모델을 만드는 방법과 개발방법론과 가이드라인의 지침을 제공하는 방법이 있으나 본 연구에서는 후자의 방법을 이용하였다.

2.2.1 소프트웨어 공학에서 메타포어의 이용

조합에 관한 문제를 해결하기 위해 다음과 같은 메타포어를 이용하였다.

- 통합환경 ➡ 시스템 ➡ 프로그램
- 프로세스 ➡ 도구 ➡ 모듈(타스크)

도구의 조립도를 고려할 때에 시스템을 프로그램에 도구를 모듈에 대응시키는 것이 유효하다. 왜냐하면, 시스템에서 도구의 밀도에 관한 문제는 프로그램과 모듈의 관련에 비교할 수 있기 때문이다.

프로세스와 도구에는 다음과 같은 3가지의 밀접한 관계가 있다.

① 하나의 프로세스에 대해 대응하는 도구가 없는 경우
프로세스에 대응하는 적합한 도구가 없어 도구를 이용하여 프로세스를 지원할 수 없는 경우이다.

② 하나의 프로세스에 대해 하나의 도구가 대응하는 경우
프로세스 자료 취득의 관점에서 바람직하다. 앞장에서 고려한 도구의 상태를 이용하여 프로세스 정보를 취득할 경우가 그렇지 않은 경우에 비해 자세한 자료를 수집할 수 있다.

③ 하나의 도구에 대해 대응하는 프로세스가 여러 개인 경우
앞 장에서 고려한 것처럼, 도구내에서 종료하기 때문에 조합시키는 관점이나 자료 취득의 관점에서 바람직하지 않다.

2.2.2 프로세스 주도의 지원환경 구축의 패스

도구는 개발 프로세스를 지원하기 위해서 선택되고 조합되며 도구를 사용함으로써 개발 프로세스도 변화한다. 대표적인 사례를 보면 내부적으로 사용하고 있는 설계 자료를 IPSE로 일괄 관리함으로써 종이로 된 문서의 필요성이 감소하였으며 리뷰 시간이 그에 따라 증가하고 있다.

프로세스 주도의 분석과 설계가 [적절히] 행해진다면 2회째 이후의 분석과 설계에 대해서 큰 영향을 주지 않는다. 프로덕트 주도로 고려했을 때 위에서 기술한 것처럼 어떤 문서가 불필요하게 된다면 큰 변경이 뒤따르게 된다. 그러나, 프로세스에 발생된 변경은 예를 들면 자료 구조의 설계에 의해 고객에게 제출할 때 종이에 출력하거나 그 프로세스가 소비하는 시간이 짧아질 뿐이고 프로세스 구조자체의 변경은 적어진다. [적절히]의 의미는 일반적으로 각각의 개발 프로세스를 대상으로 하는 도메인과 동시에 조직이나 시간과 같은 자원상의 제약에 의해 달라지고 있다. 대상으로 하는 도메인의 제약과 별도로 자원상의 제약에 얽매이지 않고 고려한 프로세스는 일정한 메타라고 하며 즉, [적절히]라고 하는 것은 이 메타 프로세스를 고려한다는 것이다. 이것은 우리가 경험하는 문서에 쓰여 있는 시간이 적다는 것과 더불어 리뷰를 충실하게 했다는 것을 나타내고 있다.

3. 환경통합화를 위한 도구의 분석과 설계 방법

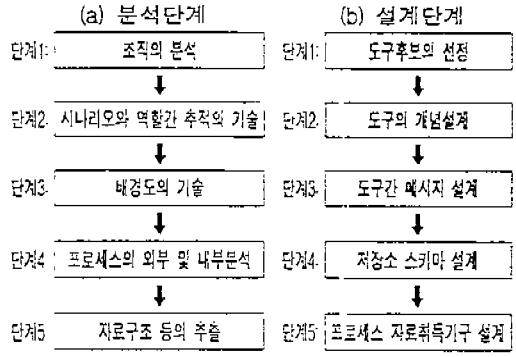
환경통합화를 위한 도구의 분석과 설계 방법에 대해 제안하기로 한다. 본 연구에서의 분석 방법은 소프트웨어 프로세스를 대상으로 한다. 설계된 도구는 프로세스를 지원하며 프로세스와 도구간의 1대 1 대응을 고려한다.

프로세스는 같은 목적을 갖는 작업의 집합으로 정의될 수 있으며 다시 프로세스 단계로 분해할 수 있다. 단순히 프로세스라고 할 때 그 프로세스 사이에 계층구조는 없으며 몇 개의 프로세스 집합을 추상 프로세스라고 부른다. 계층구조는 프로세스와 프로세스 단계 간에만 나타나고 프로세스 간에는 나타나지 않는다. 프로세스와 프로세스 단계 간의 차이는 병행성에 관련된다. 즉, 각 프로세스는 병행으로 동작할 수 있지만 각 프로세스 단계는 연속적으로만 동작할 수 있다.

3.1 도구의 분석

정밀한 도구를 설계하기 위해서 최초로 (그림 1)과 같은 분석절차를 소프트웨어 프로세스 전체에 대해 단계적으로 실시한다. 프로세스의 해석에서는 프로세스의 제거나 시간개념이 중요하므로 결국, 최초의

해석으로서 RTSA를 이용하는 것은 적절하다. 그러나 객체지향에서 [캡슐화]의 개념은 인터페이스를 명확히 하고 불필요한 액세스를 찾아 제거한다는 점에서 중요하다. 다시 말해 자료 형을 중시하는 면은 범용적으로 이용되는 저장소 스키마의 설계에 있어서 중요한 개념이다.



(그림 1) 도구의 분석과 설계 절차
(Fig. 1) Analysis and Design Process of Tool

(1) 단계 1 : 조직의 분석

우선, 해석 대상 프로세스를 실행하는 조직에 대해 분석한다. 각 역할과 그에 대한 인원수 등을 관리계층(관리자, 담당자 등)과 함께 기술한다.

(2) 단계 2 : 시나리오 및 역할간 추적도의 기술

시나리오는 소프트웨어 개발에서 각 역할이 하는 활동을 순차적으로 기술한 것이다. 시나리오는 분석 대상에 대한 비형식적인 정의이고 분석을 진행할 때 보조로 이용하며 최종적으로 작성된 분석 모델의 정당성을 확인할 때 사용하게 된다. 다음에 이 시나리오를 이용하여 대상 조직의 분석으로 얻어진 역할간의 추적도를 기술한다. 이 추적도는 분석 초기단계에서 역할을 확인하거나 프로세스가 실행하는 활동을 확인하기 위해 이용된다.

(3) 단계 3 : 배경도(context diagram)의 기술

배경도는 통합환경상의 대상으로 하는 소프트웨어 프로세스 전체가 어떠한 입력이나 출력을 외부조직(역할)이나 다른 시스템과의 사이에 실행하는가를 보이는 것이다. 최종적인 대상이 되는 소프트웨어 프로세스가 규모가 크고 복잡한 것일 때에는 소프트웨어 프로세스 전체를 서브 프로세스로 분해하게 된다.

(4) 단계 4 : 프로세스의 외부분석과 내부분석

외부분석과 내부분석은 배경도를 변환도로 분할하는 것으로 변환 도중에 자료변환 부분은 분석 초기단계에서는 추상 프로세스를 나타내고 있다. 즉, 분석을 진행하고 보다 상세한 변환도를 작성하여 프로세스 외부분석도를 만든다. 이 레벨에서의 제어 변환부분은 각 프로세스가 어떻게 제어되는가를 보이고 있다. 자료 흐름은 도구가 실현해야 할 기능이 필요로 하는 입력 자료 및 출력 자료를 나타내고 스토어는 저장소에 보존되는 자료를 나타낸다. 제어 흐름은 프로세스 통합을 위해 필요한 메시지의 흐름이다. 위에서 나타낸 프로세스로서의 각 변환 부분을 다시 분해하여 프로세스 내부분석도를 얻으며 이 프로세스 내부분석도에는 각 프로세스의 상세한 프로세스 스텝이 기술되어 있다.

상세화된 변환도가 프로세스 외부분석도인지 내부분석도인지의 구분은 먼저 프로세스의 정의에서 보인 병행성에 관한 문제 이외에 또 하나가 있다. 그것은 조직이 그 변환도에 나타난 변환 부분을 프로세스로하여 처리할 필요가 있는가의 여부이다. 프로세스 외부분석도에 나타난 제어변환 부분은 관리자 및 자동화된 IPSE 내의 프로세스 매니저가 제어해야 할 내용을 보이고 있다. 한편, 프로세스 내부분석도에서 제어변환 부분은 프로세스 내부에 있는 프로세스 스텝의 제어이다.

(5) 단계 5 : 자료 구조 등의 추출

다음에 자료사전, 이벤트 리스트, 변환사양을 작성하여야 한다. 자료사전으로부터 E-R(Entitiy Relationship)도를 작성하고 저장소에 저장될 가능성이 있는 자료의 구조를 명확히한다. 최초에 각 자료저장소의 자료에 대해 정의와 분해를 한다. 자료는 실체의 후보가 되고 그 구성요소는 속성의 후보가 된다. 또, 자료 변환부분의 변환사양에 따라 그 변환부분이 취급하는 자료에 대해 자료저장소처럼 실체와 속성의 후보를 고려할 수 있다. 자료저장에 의해 작성한 E-R도

와 변환부분에 따라 작성한 E-R도를 비교함으로써 최종적인 E-R도를 정한다. 이 E-R도는 설계단계에서 저장소의 스키마를 설계할 때의 기초가 된다. 이벤트 리스트는 프로세스의 기동과 종료에 관계하고 변환사양은 그 프로세스를 지원하는 도구의 기능을 보인다.

3.2 도구의 설계

(1) 단계 1 : 도구 후보의 선정

각 프로세스는 도구에 의한 자동화의 가능성을 가지고 있고 도구의 후보가 될 수 있다. 각 프로세스로부터 도구의 후보를 추출하는 작업은 〈표 1〉과 같은 기준을 이용한다.

〈표 1〉 도구의 후보를 추출하기 위한 기준
〈Table 1〉 standard for extract candidate of tool

점 검 사 항	처 리 기 준
프로세스는 병행으로 동작하는 활동을 포함하고 있지 않은가?	포함하고 있을 때는 다시 분해
현 시점에서 도구화 가능한 프로세스인가?	전체를 도구화하기 곤란하면 프로세스에 대한 분해를 고려
기존 도구를 사용할 수 있는가?	사용하 수 있다면 캡슐화를 검토
유사한 프로세스는 없는가?	유사 프로세스는 공통의 도구로 지원할 것을 검토

상기 기준을 적용하였을 경우 최종적으로 남은 것이 도구의 후보이다. 그리고 단계 2의 도구 개념설계에 의해 도구가 가져야 할 기능과 도구간의 인터페이스를 결정한다.

(2) 단계 2 : 도구의 개념설계

도구의 개념설계에서는 프로세스간의 병행성에 착안하여 도구의 주기능/협조관계/인터페이스 등을 정한다. 도구의 주기능은 프로세스 내부분석도를 기초로 설계되고 도구의 협조관계/인테페이스는 프로세스 외부분석도를 기초로하여 설계한다.

도구의 개념설계에서 중요한 것은 프로세스간의 병행성으로, 프로세스간에 병행성이 있는 것처럼 그 프로세스를 지원하는 도구간에도 병행성이 있다.

병행성중의 하나는 정보자원의 병행성으로서 동일 자료 테이블을 여러 사람이 작성하는 경우가 이것에 해당한다. 대상이 되는 정보는 영속적인 자료, 즉 저장소에 보존되는 자료이고, 단순한 액세스권에 대해서 경합상태는 해결된다. 또 하나는 새로운 타입의 병행성으로 CSCW(Computer Supported Cooperative Work)에 관한 것이다. 예를 들면, 동일한 DFD를 여러 사람이 동시에 작성하는 경우가 이것에 해당한다. 전자와 비교할 경우 서로의 조작 간격이 짧고 조작시의 대상은 통상 비영속적 자료라는 특징이 있다.

본 설계 방법에서는 상기의 병행성을 가지는 프로세스를 액티브 프로세스라고 하고 그중 도구화 가능한 것을 액티브 도구라고 한다. 또, 병행성을 고려할 필요가 없는 프로세스를 패시브 프로세스라고 부르고 도구화 가능한 것을 패시브 도구라고 부른다. 도구 개념설계 때에 작성되는 도구 구조도에서 액티브 도구는 평행사변형이고 패시브 도구는 직사각형으로 표현하였다. 각 도구내의 평행사변형 및 직사각형은 도구의 주기능을 표현하고 있다.

예로서 여기에서는 프로세스 모델화 방법론을 비교하기 위해 ISPW6에서 사용된 예제를 사용하였다[6]. 이 예제는 소프트웨어 변경 프로세스를 다루는 것으로 설계, 코딩, 단체 테스트 및 관리에 관련되어 있다. (그림 2)에 보인 것은 그 안에 소스 코드의 수정에 관련된 부분의 도구 구조도이다.

분석도 내의 자료저장소의 취급시에 주의가 필요하나 자료저장은 일반적으로 프로세스라고 간주되지 않는다. 그러나 몇가지 이유로 자료저장에 관한 프로세스는 정보은폐 도구((그림 2)의 이중선 평행사변형 부분)로 치환하고 다른 액티브 도구로부터 자료저장소가 가지는 자료 구조를 숨기는 것을 생각할 수 있다.

첫째 이유는 그 구조에 대한 부주의한 액세스를 막기 위해서이고, 둘째 이유는 작성/갱신/소거하는 기본적인 조작으로부터 도구를 해방하기 위해서이다. 일반적으로 저장소에는 복잡한 자료 구조를 추상화하는 능력이 있다. 그러나 이와 같은 기본적인 조작을 정보은폐 도구에 위임함으로써 도구가 의존하는 스키마 및 인터페이스가 단순화된다. 또, 다음에 기술하는 것처럼 프로세스의 자료 취득에 관한 문제를 보다 쉽게 취급할 수 있다.

2.1절에서 기술한 프로세스의 자료 취득에 관한 고

려를 도구의 개념설계에서 진행한다. (그림 2)에서 이 자료 취득에 관련된 부분은 액티브 도구 및 정보은폐 도구로 나타낼 수 있다. 액티브 도구에서 그 내부에 p라고 하는 기호와 함께 평행사변형으로 나타낸 부분은 도구의 상태를 취득하고 프로세스 매니저에게 송신한다. 정보은폐 도구에서는 객체에 대한 기본조작 및 내부에서 가지고 있는 객체 특성을 송신한다. 또, 이것에 의해 도구가 가지는 프로세스 정보수집 기구가 그림 중에서 명시화된다.

(3) 단계 3 : 도구간 메시지 설계

도구간에 전달되는 메시지의 설계를 한다. 도구간 메시지로서는 「저장소」와 「통지」의 2가지 형태가 있고 이 중에서 선택한다. 리퀘스트(request)는 다른 도구에 대해 기능의 실현을 요구하는 메시지로서 일반적으로 동기형의 통신이다. 통지는 다른 도구에 대한 이벤트의 송부이며 이 경우는 통상 비동기형의 통신이다.

(그림 2)의 도구 구조도에 보인 예에서 코드편집 매니저와 컴파일 도구는 쌍방향 메시지이고 리퀘스트로써 실현된다. 메시지 큐에서 보이는 설계수정으로부터 코드 편집 매니저로의 단순한 서비스 요구는 통지이다.

(4) 단계 4 : 저장소 스키마 설계

스키마 설계에서는 E-R 표현을 스키마로 변환한다. 분석 단계에서 추출된 모든 E-R도에 대해서 동일한 엔터티를 결합하는 것이고 하나의 큰 E-R도를 만든다. 이 과정에서 몇가지 저장소의 제약을 고려할 필요가 있다. 예를 들면, PCTE의 경우 E-R에 있는 3항 관계는 2항 관계의 링으로 변환한다. 객체형의 내부 개념 스키마로부터 수입이 가능한 것이 있다면 그것을 이용한다. 또, 기존의 형으로부터 계승할 수 있는지의 유무에 대해서도 살펴본다.

그리고 작성자의 시나리오나 역할간 추적도를 이용해서 도구 구조도에 보인 병행처리 등을 고려하여 액세스 관리, 안전성에 대해서도 검토한다.

(5) 단계 5 : 프로세스 자료취득 기구의 설계

2.1절에서 기술한 프로세스 자료 취득법 안에 먼저 도구의 상태를 이용한 측정에 대해 고려한다. 도구의 상태는 그 도구가 현재 어떤 상태에 있는지에 대해 다음의 형식(M1)으로 통지한다.

M1 : (프로세스명, 조작자, 상태, 액션 정보)

프로세스명은 도구가 관여하는 프로세스의 이름이고, 상태는 기동중인가의 여부이며 일정시간 이상 키보드없이 하는 마우스 이벤트가 발생하지 않는 경우에는 대기상태로 판단한다. 액션정보는 하나 이상의 (액션명, 실행회수)의 조합이다. (그림 2)의 Code_Editor의 경우 이 조합은 7개 있다.

정보은폐 도구는 아래에 표시한 형식(M2)에서 M1과 함께 객체의 상태/특성에 관한 자료를 송신한다.

M2 : (객체 종류(형), 생성시간, 갱신회수, 메트릭스 정보)

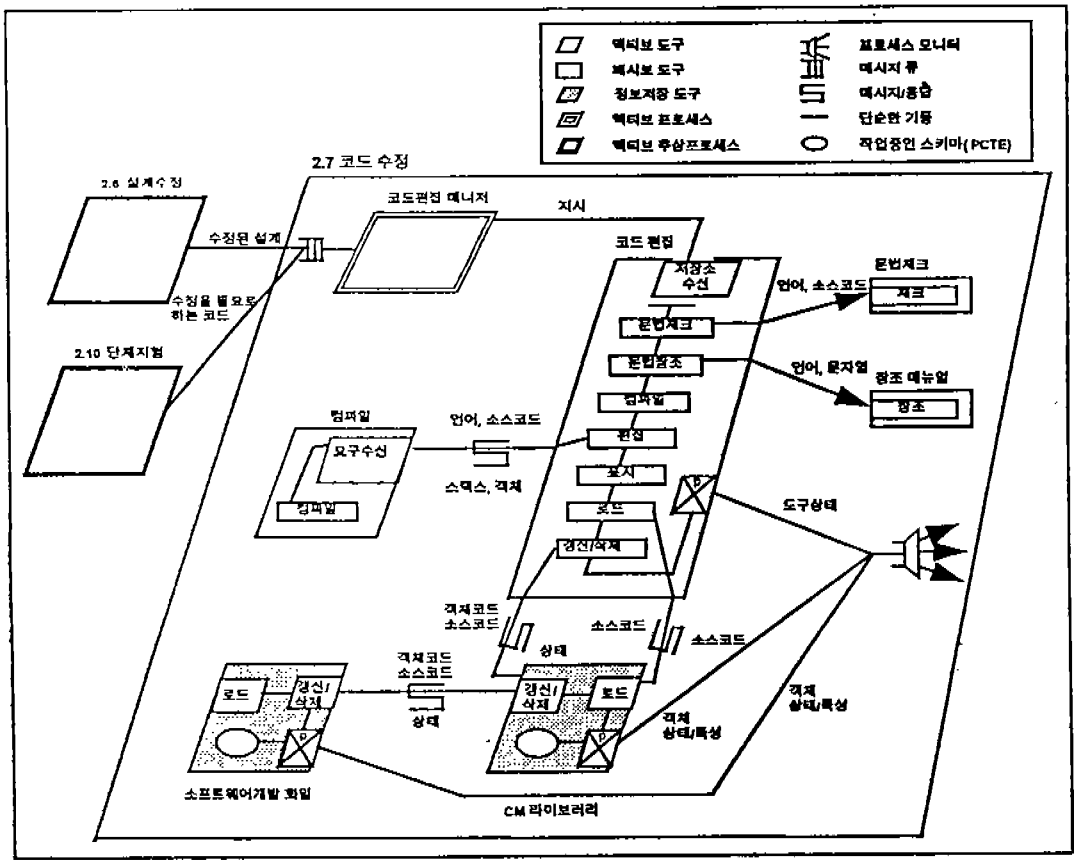
메트릭스 정보는 1개 이상의 (메트릭스명, 메트릭스)의 조합이고, 자료의 특징(형)에 의해 정한다. 예를 들면 DFD의 경우에는 Class weight[9]를 프로그램 코드의 경우에는 McCabe의 Cyclomatic Complexity[10]를 이용할 수 있다. 여기까지 스키마 설계로 설계된 자료의 특징에 대응하여 적절한 메트릭스 취득방법을 검토할 필요가 있다.

3.3 도구의 결합도와 응집도

설계에서 프로세스의 분할이 불충분한 경우도 있고 도구가 필요이상으로 상세한 밀도를 가지는 경우도 있다. 여기에서는 구조화 설계에서 이용되는 개념 즉 결합도와 응집도[11]를 본 연구에서 도구통합화를 위한 도구 설계방법으로 확장하여 제안하였다. 도입한 새로운 기준은 도구간의 의존관계를 나타내는 도구결합도와 도구내 각 기능의 연결 강도를 나타내는 도구응집도이다.

(1) 도구의 결합도

구조적 분석과 설계에서의 모듈간 결합도의 개념을 도구간의 개념으로 확장하여 도구간의 연관성을 나타내는 척도로서 〈표 2〉에 도구간 결합도의 유형을 제안하였다. 도구간 결합도는 모듈간 결합도와 마찬가지로 결합 강도가 낮을수록 좋은 결합도를 나타낸다.



(그림 2) 소스코드의 수정에 관련된 부분의 도구구조도의 예
(Fig. 2) Example of tool structure chart of a part about correction of source code

(2) 도구의 응집도

구조적 분석과 설계에서의 모듈을 구성하는 요소간의 연관성 정도를 나타내는 응집도의 개념을 확장하여 도구내의 구성요소간의 연관성을 나타내는 척도로서 〈표 3〉에 도구간 응집도의 유형을 제안하였다. 도구간 응집도는 모듈간 응집도와 마찬가지로 응집 강도가 높을수록 좋은 결합도를 나타낸다.

4. 도구의 통합적합성에 대한 메트릭스

여기에서는 3장의 방법에 대해 분석과 설계된 도구가 통합을 위한 도구로서 적절한가를 판단하기 위한 평가 메트릭스를 제안한다. 메트릭스는 도구가 상세한가에 관한 정밀도, 도구 조합의 용이성 및 환경의 도구화율의 3가지의 평가지표로서 이들은 분석과 설계법에서 기술한 변환도 및 도구 구조도를 이용하여 계산할 수 있다.

4.1 정밀도(Granularity)

도구의 세밀함의 정도인 정밀도(G)는 그 도구가 얼마만큼의 프로세스를 지원하고 있는가에 대해 다음 식과 같이 정의할 수 있다.

$$G=\frac{1}{\text{도구가 지원하고 있는 프로세스의 수}}$$

도구가 지원하고 있는 프로세스의 수가 1일 때, 정

〈표 2〉 도구간 결합도의 유형과 대처 방법
〈Table 2〉 Type of coupling between tools and solution method

정 의	강도	의 미	대처방법
①도구의 단순 결합도	결합도가 낮다	단순히 어떤 도구가 다른 도구를 호출하는 것으로 도구가 기동된 후 다른 도구는 병행으로 동작한다.	그대로 둔다.
②도구 국소 자료 결합도		어떤 자료의 교환을 직접 도구간 메시지로 하는 것을 전제로 하는 결합이다. 메시지의 논리적인 사이즈가 큰 경우 즉, 메시지가 복잡한 자료 구조를 갖는 경우 자료 구조가 불투명해지고 그 도구를 적용할 수 있는 범위는 한정된다. 또, 자료 양이 큰 경우에는 명확하게 저장소를 이용하여 도구간의 자료의 교환을 하는 것이 바람직하다.	
③도구의 제어 결합도		도구가 전달하는 자료에 의해 자료를 전달한 도구의 거동이 변화하는 경우의 결합이다. 일반적으로 프로세스의 관리를 하고 있는 도구와 그 관리하에 있는 도구와의 관련이 제어결합이 된다. 프로세스 관리도구 이외의 것과 제어결합을 맺는 것은 바람직하지 않다.	일단 통합하여 재분할을 검토한다.
④도구의 공통 결합도		불휘발성의 자료 영역을 개입시켜 도구간에 자료 교환을 하는 경우이다. 저장소를 경유하지 않고 공유 메모리를 이용하여 공통결합하고 있는 경우 자료 구조의 의미가 불명확해지기 쉬우므로 피해야 한다.	일단 통합하여 재분할을 검토한다.
⑤도구의 내용 결합도	결합도가 높다.	어떤 도구가 일련의 기능을 실현할 때에 다른 도구의 실행을 전제로 하는 결합이다. 이 경우, 도구간에 불필요한 의존관계가 생긴다. 다음에 설명하는 도구 응집도의 관점에서 문제가 없는 한 그것들은 하나의 도구라고 하는 것이 바람직하다.	1개의 도구로 통합한다.

밀도 G는 1이고 IPSE용 도구로서 적절하다고 볼 수 있다. 하나의 도구로 다수의 프로세스를 지원하는 경우는 도구의 분할이 적절하지 못하다는 것을 의미하며 도구가 구현하고 있는 기능간에 직접적으로 관련이 없는 경우이다. 이때 G의 값은 1보다 작아진다. 예를 들면, 「도구의 우연적 응집도」를 가지고 있는 도구가 이에 해당된다.

다음에 도구와 프로세스간의 관계인 수평도에 대해 고려한다. 일반적으로 도구는 수평형 도구와 수직형 도구로 나눌 수 있다. 수평형 도구는 복수의 추상 프로세스에서 사용되는 버전관리나 문서지원 등의 도구이고 수직형 도구는 특정의 추상 프로세스에만 사용되는 도구이다. 수평도 H는 그 도구가 사용되고 있는 추상 프로세스의 수로 정의하고 이 수평도를 고려한 수정정밀도 G^*는 다음식으로 표현된다.

$$G^* = H \times G$$

이 수정정밀도 G^*에서 값이 1보다 작을 때의 의미는 정밀도의 경우와 동일하다. 값이 1보다 클 때 그 도구는 수평형이고 그 값은 수평형의 정도를 보인다.

4.2 조합의 용이성(Ease of Combination)

조합의 용이성 EC는 어떤 특정한 도구를 어느 정도 용이하게 다른 도구와 조합시킬 수 있는지의 수준을 파악하는 것이다.

이 조합의 용이성은 다른 도구간의 인터페이스의 수에 의해 다음 식과 같이 정의한다.

$$EC = 1/(Cdi \times NUM(Din) + Cci \times NUM(Cin) + Cdo\ NUM(Dout) + Cco \times NUM(Cout))$$

Din : 자료 인터페이스
Cin : 제어 인터페이스
Dout, Cout : 출력 인터페이스
NUM() : 해당 인터페이스의 수
Cdi/o, Cci/o : 양의 값을 갖는 계수
(PCTE 같은 표준 저장소를 이용하지 않는 경우, Cd의 값은 커진다)

도구간에 자료의 교환이 없고 단순히 기동되기만 하는 도구의 조합 용이성은 Cc가 1인 경우 1이고 전형적인 패시브 도구임을 표시하고 있다. 인터페이스 수가 증가함에 따라 이 조합의 용이성 EC는 감소한다.

4.3 도구화율(Tooling Rate)

프로세스 분석도에서 고찰한 프로세스의 내부중에서 도구화할 수 있는 것을 조사하는 것으로 도구화의

〈표 3〉 도구간 응집도의 유형과 대처 방법

〈Table 3〉 Type of cohesion between tools and solution method

정 의	강도	의 미	대처방법
①도구의 단순 기능적 응집도	응집도가 높다.	하나의 도구가 단일 프로세스의 기능을 실현하고 있으며 사용자와 도구 사이에 상호작용이 없는 도구이다. 예를 들면, 문법 체크와 같은 배치적인 처리를 하는 것이 그것에 해당한다.	분할할 필요가 없다.
②도구의 순차적 응집도		사용자에게 도구가 어떤 일정한 절차에 따르는 입력을 요구하는 경우이다. 적절히 입력을 안내하는 것은 효과적이지만 불필요한 절차를 강요하는 것은 피해야 한다.	분할할 수 있는 가능성을 검토해 보아야 한다.
③도구의 논리적 응집도		도구가 논리적으로 동일한 종류의 몇 가지 기능을 포함하고 있고 그것들이 동시에 사용되지 않는 경우이다. 기본적으로 이들 도구는 분할되어 4장에서 기술한 도구의 카테고리에 따라 등록되는 것이 바람직하다.	기본적으로 분할되어 4장에서 기술한 도구의 카테고리에 따라 등록되는 것이 바람직하다.
④도구의 우연적 응집도	응집도가 낮다.	그 도구가 실현하고 있는 기능간에 직접 관련이 없는 경우, 혹은 관련이 적은 경우이다. 예를 들면 DFD로부터 mailer를 기동하는 경우에 mail 기능을 DFD 편집기에 추가하고 있는 경우가 이것에 해당한다.	분할되는 것이 바람직하다.

정도는 도구의 수정 정밀도 G^*를 이용하여 다음의 식으로 정의한다.

$$TR = \frac{\sum_{i} G_i^*}{\text{프로세스 수}}$$

도구 수를 대신하여 각 도구의 수정 정밀도 G^*를 사용함으로써 정밀도 도구를 적용한 경우 TR은 그렇지 않은 경우에 비해 작아지지 않는다. 또, 수평형 도구는 복수의 프로세스에서 사용되기 때문에 위 식을 이용하지 않는 경우에 TR의 값은 작아지지만 G^*를 사용함으로써 올바른 TR을 산출할 수 있다.

4.4 고찰 및 평가

개발환경은 한 번 구축된 것으로 끝나는 것이 아니라 성능향상을 위해 계속적으로 유지 보수가 이루어져야 한다. 개발환경을 구축하기에 적합한 좀더 향상된 기능 및 성능을 갖춘 도구들이 계속해서 개발될 것이고 이미 구축되어 있는 개발환경도 이에 따라 지속적으로 유지 보수가 필요하게 된다. 이때 유용하게 사용할 수 있는 개념이 도구의 결합도와 응집도에 대한 개념으로 통합개발환경에 대한 유지 보수를 용이하게 하기 위해 필요하다. 만일 하나의 도구가 다수의 프로세스를 지원하고 있다면 이 도구를 대치할 수 있는 좀더 향상된 하나의 도구를 찾는 것은 매우 어려운 일이 될 것이며 기능이 좀더 분할된 도구로써 대치한다면 유지 보수 비용의 불필요한 낭비를 초래할 수도 있다. 왜냐하면 분리된 기능들중 일부는 이미 기존의 도구에서도 만족할 만한 수준으로 지원되고 있을 수도 있기 때문이다. 그러므로 개발환경을 향상시키기 위한 유지 보수시 뿐만 아니라 처음 개발환경을 구축할 때부터 도구의 응집도와 결합도의 개념은 유용하게 활용될 수 있다.

그리고, 기존에 구축되어 있는 개발환경에 대한 평가를 위해 앞서 제안한 평가지표들을 이용할 수 있

다. 도구의 정밀도는 1 이상이 좋으며 정밀도의 값이 1보다 작을 때는 도구의 분할이 적절하지 못함을 의미한다. 조합의 용이성은 1에 가까운 것이 바람직하다. 이것은 도구 개발자가 도구를 개발할 때, 또는 사용자가 도구를 구입할 때의 지표가 된다. 이 도구화율은 사용자 사이트에서 환경구축자가 자신의 환경을 정기적으로 평가할 때 지표로 이용할 수 있다.

현재 도구들을 결합하여 통합환경을 구축하기 위한 분석/설계에 관한 연구는 별로 없는 실정이다. 그러나 많은 일반 시스템에 관한 분석/설계법이 연구되어 있고, 본 연구에서 보인 분석/설계법에서도 기존의 분석/설계법의 개념이나 도식을 이용하고 있다.

도구 개념도에 관한 도식에는 CODARTS[8]에서 사용되고 있는 것을 일부 이용하고 있다. CODARTS는 병행처리의 설계를 목적으로 하고 있고 병행동작하는 행동의 기술이 가능하다.

SADT도로부터 PCTE 스키마를 설계하는 방법에 대해서는 [7]에서 자세하게 기술되어 있다. ER도로부터 PCTE 스키마를 작성할 때에 [7]에 있어서 스키마 설계에 관한 의논은 본론에서 보인 기법과 함께 사용할 수 있다.

본 연구에서의 기본적인 사고 방식은 도구의 정밀도화이다. 결국, 도구가 실현하는 기능은 단일이고, 도구단위로 제공 및 이용이 가능하여야 한다.

5. 결 론

본 연구에서는 통합 소프트웨어 개발환경을 구축하는 경우에 발생할 수 있는 2가지 문제점에 대해 고려하였다. 첫째는 도구의 밀도(Granularity)에 관한 문제로서 사용자는 만족할 만한 성능을 얻기 위해 다양한 기능을 갖는 도구들을 통합하게 되는데 여러 기능을 하나로 통합하는 조립 과정을 거친 도구는 일반적으로 다른 도구와의 인터페이스가 큰 문제로 대두될 수 있다는 점과 둘째는 프로세스에 관한 자료를 수집하고자 할 때 도구를 이용한 자동적인 측정에 대해 사전에 대비하고 있지 않다는 점이다. 따라서 이와 같은 문제점을 해결하기 위해 소프트웨어 개발환경의 분석과 설계절차를 제안하였다. 이 기법은 프로세스 분석을 개시점으로 하는 것이고 본 기법 중에서 프로세스 자료취득 기구의 설계에 대해서도 검토하였다. 그리고 동시에 적절히 분석과 설계를 하기 위한 기준 및 환경을 평가하기 위한 평가 메트릭스를 제안하였으며 소프트웨어 개발환경을 통합하기 위해 필요한 도구의 선택을 위해 구조적 분석과 설계에서의 모듈에 관련된 결합도와 응집도의 개념을 확장하여 도구에 관련된 결합도와 응집도의 개념으로 변환하였다. 통합 소프트웨어 개발환경에 대한 평가를 위해 제안한 평가 메트릭스를 통해 개발환경을 구성하고 있는 도구에 대해 평가하고 개발 환경이 적절하게 구성되어 있는지에 대해 평가할 수 있다. 또한, 환경구축자는 자신의 환경을 정기적으로 평가함으로써 보다 나은 개발환경을 구축하는데 도움을 받을 수 있을 것이다.

향후 연구과제로서는 사용자에 대해 정밀도 도구 및 프로세스 지향의 환경을 제공하고 보다 다양한 환경이나 프로세스에 본 연구를 적용할 수 있도록 할 예정이며 도구 모델이나 분석과 설계법 및 평가기준에 대한 검증을 계속해 나갈 예정이다.

참 고 문 헌

[1] ECMA, "Reference Model For Frameworks of Software Engineering Environment", Nist Special Publication 500-21 Technical Report ECMA TR/55 2nd ed., Doc. 1991.

[2] Harrison W., Ossher H., "PCTE SDSs For Modeling OOTIS Control Integration", Proceedings of the PCTE '93, 1993.

[3] Paulk, M., et al. "Capability Maturity Model", Ver. 1. 1, IEEE Software, Vol. 10, No. 4, 1993.

[4] Curtis, B., et al., "Process Modeling", Commun. ACM vol. 35, No. 9, pp. 75-90, 1992.

[5] Ward, P., Mellor, S., "Structured Development for Real-Time Systems", Yourdon Press, 1985.

[6] Kellner, M., et al., "Software Process Modeling Example Problem", ISPW6, pp. 19-29, 1990.

[7] Bremeau, C., Thomas, I., "A Schema Design Method For PCTE", Proceedings of the PCTE '93 Conference, pp. 117-137, 1993.

[8] Gomma, H., "Software Design Methods for Concurrent and Real-Time Systems", Addison-Wesley,

1993.

[9] DeMarco, T., "Controlling Software Projects", Yourdon Press, 1982.

[10] McCabe, T. J., "A Complexity Measure", IEEE Trans. Software Eng., pp. 308-320, 1976.

[11] Page-Jones M., "The Practical Guide to Structured Systems Design", 2nd ed., Prentice-Hall, 1988.

[12] Thomas, I., Nejmeh, B., "Definitions of Tool Integration for Environments", IEEE Software, Vol. 9, No. 2, 1992.

[13] 이하용, 이용근, 박정호, 양해술, "소프트웨어 복잡성 측정시스템의 설계 및 구현", 한국정보처리학회, 정보처리논문지, Vol. 2, No. 3, 1995. 5.

[14] 양해술, 이용근, 이하용, "프로세스모델 기반 개발방법과 프로세스의 평가", 한국정보과학회, 정보과학회지, Vol. 13, No. 9, 1995. 9.

[15] 양해술, 이용근, 허태경, "소프트웨어 프로젝트 관리에서의 품질보증 시스템의 프로세스 기술방식", 한국정보처리학회, [정보처리], Vol. 1, No. 3, 1994. 9.

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 전자계산실 시스템분석장교 근무

1986년~1987년 日本 오사카대학교 객원연구원

1993년~1994년 한국정보과학회 학회지 편집부위원장

1980년~1995년 강원대학교 전자계산학과 교수

1994년~1995년 한국정보처리학회 논문지편집위원장

1994년~현재 한국산업표준원(IIS) 이사

1995년~현재 한국소프트웨어품질연구소(INSQ) 소장

관심분야: 소프트웨어공학(특히, S/W 품질보증과 품질평가, 품질감리, 품질컨설팅, OOA/OOD/OOP, CASE, SI), 소프트웨어 프로젝트관리.

이 하 용

1993년 강원대학교 전자계산학과 졸업(학사)

1995년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)

1995년~현재 강원대학교 대학원 전자계산학과 박사과정

1996년~현재 경희대학교 전자계산공학과 강사 한국소프트웨어품질연구소 전임연구원

관심분야: 소프트웨어공학(특히, S/W 품질보증과 품질평가 및 품질감리 객체지향 프로그래밍, 객체지향 분석과 설계, CASE).