소프트웨어 프로세스의 작업흐름 분석과 명세정의 방법

양 해 술†

요 약

비지니스 프로세스를 근본적으로 개선하는 접근방법으로 BPR이 주목받고 있으며 최근에는 비지니스 프로세스의 속도 향상과 기술자의 생산성 향상을 목적으로 작업흐름 관리와 자동화에 특히 관심이 집중되고 있다. 소프트웨어공학 분야에서도 프로세스에 초점을 맞춘 접근방법으로 소프트웨어 프로세스 공학이 주목받기 시작하고 있으며 소프트웨어의 생산성 및 품질 향상과 개발기간의 단축을 위해 소프트웨어 프로세스를 근본적으로 개혁하기 위한 방안으로 SPR에 대한 연구가 진행되고 있다. 따라서, 본 연구에서는 소프트웨어 프로세스 리엔지니어링 방법론 확립의 일환으로서 소프트웨어 프로세스의 작업흐름 관리 시스템 구축을 위한 작업흐름 분석과 설계방안을 구축하였다. 즉, SPR 프로세스와 소프트웨어 프로세스의 구조를 모델화하는 과정과 소프트웨어 프로세스의 구조를 분석하여 소프트웨어 프로세스의 작업흐름 모델을 구축하는 작업흐름 분석 기법과 작업흐름 소프트웨어의 명세정의 기법에 대한 방안을 강구하였다.

Work-flow Analysis and Specification Definition Method of Software Process

Hae-Sool Yang†

ABSTRACT

BPR(Business Process Reengineering), a radical improvement approach of business process, has been paying attention, and work-flow management and automation has been concentrated on progress velocity of business process and productivity of engineers. In software engineering, software process engineering which focus to process begin to be watched, and for the purpose of software productivity and quality progress and reduction of development term, study on SPR(Software Process Reengineering) is being progressed.

In this paper, made workflow analysis and design method for construction of work-flow management system of software process to stand firm process reengineering methodology. In other words, we studied modeling process methods for SPR process, and software process structure and workflow analysis method which construct software process workflow model and specification definition method of workflow software.

1. 서 론

비지니스 시스템 분야에서 비지니스 프로세스를 근본적으로 개선시키기 위한 접근방법으로서 BPR(Business Process Reengineering)이 주목받고 있다[3]. 또한 최근에는 비지니스 프로세스의 속도 향상과 기술자의 생산성 향상을 목적으로 작업흐름 관리와 작업흐름 자동화에 특히 관심이 집중되고 있다. 작업흐름 관리는 비지니스 프로세스에 의존하는 전표와 문서 등의

† 종신회원 : 한국소프트웨어품질연구소(INSQ) 소장
논문접수 : 1997년 2월 21일, 심사완료 : 1998년 2월 2일

형식을 전자화하고 그 흐름을 자동 제어, 감시, 그리고 관리한다[1]. 소프트웨어공학 분야에서도 프로세스에 초점을 맞춘 접근방법으로 소프트웨어 프로세스 엔지니어링이 주목받기 시작하고 있으며 소프트웨어의 생산성 및 품질 향상과 개발기간의 단축을 위해 소프트웨어 프로세스를 근본적으로 혁신하는 방안으로 SPR(Software Process Reengineering)에 대한 연구가 시작되고 있다. 그러나 SPR에서 작업흐름 관리를 실현하기 위해서는 이론과 실제의 양면에서 연구, 개발, 적용, 평가를 시행할 필요가 있다.[7, 11, 12].

또한, 소프트웨어 개발 과정을 형식적으로 기술하게 됨에 따라 소프트웨어 개발 지원의 시험이 활발해지고 있다. 지금까지 소프트웨어 개발에서 작업 방침과 사고 방식은 명시되어 있어도 구체적인 도구 이용과 작업 순서에 대해서는 명시되어 있지 않은 경우가 많다. 그 때문에 개발자는 모듈 설계, 소스 프로그램 작성 등의 본질적인 작업이외에는 작업진행 관리, 도구 선택, 기동, 화일 관리 등을 스스로 행해야 했다.

현재, 소프트웨어의 개발작업에 따른 도구의 기능이나 메시지의 표시 등을 기술하거나 실행하기 위한 언어 PDL(Process Description Language)과 그 인터프리터를 작성하고 있다[1, 3]. 또한 이 시스템을 이용하여 여러 가지 프로세스를 기술하고 실행해왔다[6]. 지금까지 시험을 통해 도구의 기동과 윈도우 조작 등의 프로세스는 기술할 수 있지만 소프트웨어 개발에 따른 제품(문서와 소스 프로그램 등의 생성물)의 관리 작업을 기술하는 것은 매우 복잡한 일임을 알 수 있다. 이것은 각각의 제품이 여러 가지 상호관계를 가지고 있고 개발 작업중에 이들 관계가 동적으로 변화하기 때문이다.

본 논문에서는 프로세스를 형식적으로 기술해 두고 그 기술에 따라 작업을 진행하는 방식을 생각하였다. 아울러 이 기술에 따라 작업 내용에 적합한 도구를 순차적으로 기동하고 수행할 작업을 개발자에게 지시하는 시스템이 있으면 개발자는 지시된 작업에만 전념할 수 있으며 일련의 작업을 효율적으로 수행할 수 있게 된다. 이와 같은 시스템은 개발자의 부담을 경감시키고 소프트웨어의 생산성 및 품질 향상에도 효과가 있다고 생각된다.

또한 최근에 비지니스 시스템 또는 시스템 통합(SI) 분야에서 첫째, 작업흐름 프로세스의 스케쥴링 문제 둘째, 작업흐름 프로세스의 객체지향 분석의 연구에 관심을 가지면서 상기의 작업흐름 연구를 소프트웨어 프로세스에 전개하여 소프트웨어 프로세스에서 작업흐름 관리 시스템의 구축 방법론을 연구하기 위한 방안을 모색하였다. 소프트웨어 프로세스에서 작업흐름 관리의 목적과 목표는 첫째, 작업의 기동을 자동화함으로써 연속된 작업 사이에 낭비하는 시간을 배제하며 둘째, 문서베이스의 프로세스 관리를 실현하고 관리정밀도의 향상을 도모하고 셋째, 문서화를 자동화하는 것보다 리드타임(Read Time)을 단축하며 넷째, 품질관리 데이타를 자동 수집함으로써 품질관리 정밀도의 향상을 도모하는 것이다.

따라서, 본 연구의 목적은 소프트웨어 프로세스 리엔지니어링 방법론 확립의 일환으로서 클라이언트/서버 시스템형의 하드웨어 네트워크 시스템에서 구축되는 소프트웨어 프로세스의 작업흐름 관리 시스템 구축을 위한 작업흐름 분석과 기술방법을 모색하였다. 즉, 2장과 3장에서는 프로세스 기술과 산출물 관리의 중요성 및 소프트웨어의 구조와 소프트웨어 프로세스의 구조를 모델화하는 과정을 기술하였으며 제4장에서는 소프트웨어 프로세스의 구조를 분석하여 소프트웨어 프로세스의 작업흐름 모델을 구축하는 작업흐름 분석 방법에 대하여 모색하였다. 그리고 5장에서는 이 모델을 기초로 한 작업흐름 소프트웨어의 명세정의 기법에 대한 작업흐름 설계 사례 방안을 강구하였으며 끝으로 연구결과에 대한 분석과 고찰 및 연구결과와 향후 연구과제에 대해 기술하였다.

2. 관련 연구

2.1 작업흐름

작업흐름은 다수의 인원이 관련된 프로세스에서 사용된 순차적 및 동시적 순서 또는 스텝으로 정의된다[2, 3, 6, 10]. 특히, Hammer[2]는 작업흐름이 활동의 불확실성, 분산성, 추상성, 변화성, 반복성을 가지며 활동간의 협동성과 비동기성에 관한 연구를 진행하였으며 사건 지향적인 상태 변화에 대해 가시적인 형태를 제안하고 있다.

작업흐름의 관리는 Aoyama[1]와 문헌[7, 8]의 연구에서 관리목표에 따라 차이가 있지만 통상적으로 다음과 같은 관리를 수행하고 있다. 먼저, 첫 번째로 분석시에 작업흐름의 특성을 분석하는 것으로서 정적 또는 동적인 특성과 주관적이며 객관적인 특성 그리고 정

성적 또는 정량적인 특성으로 분류하고 있다. 두 번째로 설계시에는 프로세스에 대한 작업흐름을 설계하거나 현재의 효율이 낮아 재설계에 의한 작업흐름의 개선에 관련된 연구 분야를 의미한다.

그리고, 세 번째의 관리시에는 현재의 작업흐름 상태를 파악하여 성능과 효율 및 생산성 그리고 보안성에 관련된 저수준의 문제들을 관리적인 방법으로 극복하는 것을 보여 주고 있다. 끝으로 작업흐름의 자동화는 연구의 최종 목표로서 조직의 프로세스 자동화를 실행하면서 다른 기술과의 접목에 의한 자동화를 종합적으로 추진하고 있다.

2.2 프로세스의 기술

본 논문에서는 프로세스를 한 사람의 개발자가 한 대의 워크스테이션에서 수행하는 일련의 작업이라고 정의하며 팀을 이룬 복수의 컴퓨터를 이용한 개발이나 발주자와의 상호관계 등의 인적 측면에 대해서는 고려하지 않았다.

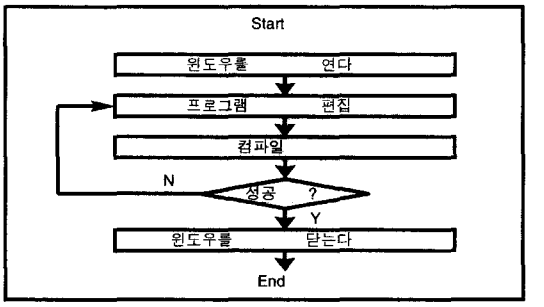
그리고 작성한 기술언어 PDL에서는 실행가능한 프로세스의 기술을 도구의 기동, 윈도우의 조작, 메시지의 표시, 개발자로부터의 응답 등의 입력계열로서 기술한다. 또한 다음 작업으로 진행해도 좋은가 또는 작업을 다시 고쳐야 할 것인가 등의 조건을 판정하고 작업의 순서를 결정하는 것이 가능하다.

일반적인 간단한 C 프로그램 개발 작업의 흐름은 (그림 1)과 같으며 이 프로세스를 PDL언어로 기술하여 실행시키면 자동적으로 새로운 윈도우가 열려 에디터가 기동하며 소스 프로그램을 작성하여 에디터를 종료하면 자동으로 컴파일한다. 컴파일에서 잘못이 발견되면 다시 프로그램 편집 단계로 되돌아간다. 이와 같이 프로세스 기술에 따라 개발작업의 지원이 가능하다.

2.3 제품 관리의 기술

개발작업에서 작성되는 문서, 소스 코드, 목적 프로그램 등을 제품이라고 한다. 제품은 종류와 상태, 상호관계, 물리적인 정보 등의 제품의 속성을 가지며 또한 생성, 편집, 표시 등의 조작이 가능하다.

(그림 1)에서 든 예는 C 소스 프로그램이 하나밖에 없는 단순한 것이다. 그러나 일반적으로 프로그램은 복수의 모듈로 구성되고 몇 개의 소스 화일, 헤더 화일로 구성되어 있다. 이와 같은 복수의 제품으로 구성되는 소프트웨어 개발을 위해서는 각각의 제품의 생성, 수정 또는 버전 관리 등의 작업을 지원해야 한다. 예를 들면



(그림 1) 간단한 C 프로그램의 개발 프로세스
(Fig. 1) Development Process of Simple C Program

어떤 소스 화일을 수정하는 경우 동시에 수정해야 하는 소스 화일은 무엇인가, 다시 만들어야 하는 객체 프로그램은 무엇인가 등의 정보를 이용하여 필요한 부분의 수정을 효율적으로 수행하고 싶은 요구가 있다. 그러나 소프트웨어를 어떤 모듈로 구성하고 어떤 화일로 분할하는가 라는 것은 개발을 시작하고 나서 결정할 사항이다. 또한 나중에 변경되는 것도 있다. 그러므로 개발작업중에 나타나는 모든 제품에 대해서 작업과 정보를 미리 기술해 두는 것은 불가능하다.

또한 프로세스의 흐름을 중심으로 한 지금까지의 기술방법에서는 제품의 관리에 대한 기술과 프로세스 흐름에 대한 기술이 혼재하므로 기술·변경을 수행하기 어려운 면이 있다. 화일의 편집 방법과 도구의 기동 방법 등 제품의 속성에 의존하는 부분을 프로세스의 흐름과 분리하여 기술해 두면 각 부분의 기술 및 변경이 용이하게 되고 재이용성도 향상된다.

3. 소프트웨어 프로세스 모델

3.1 시스템 구조 모델

본 연구에서 작업흐름 관리의 대상이 되는 것은 클라이언트/서버 시스템형의 하드웨어 네트워크 시스템에서 구축되는 소프트웨어이다[7]. 이 때 대상이 되는 소프트웨어와 하드웨어 네트워크 시스템을 포함한 전체 시스템은 (그림 2)와 같은 시스템 구조모델로 실현한다.

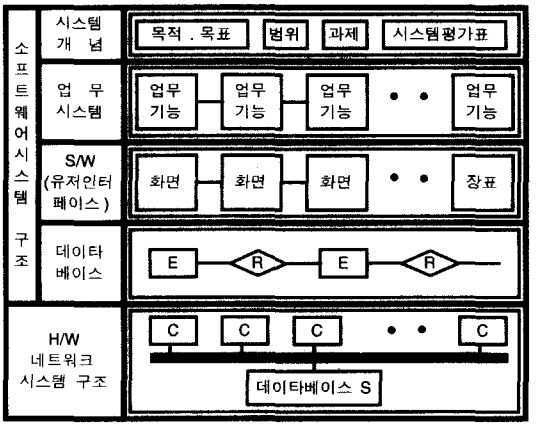
(1) 소프트웨어 시스템 구조 모델

① 시스템 개념 층 : 시스템의 목적과 목표, 범위, 시스템 평가로부터 생기는 추상적인 층이다.

② 업무 시스템 층 : 시스템 개념을 실현하기 위해 사용자가 행하는 업무기능에서 생기는 층이다.

③ 소프트웨어(사용자 인터페이스) 층 : 시스템 개념으로부터 정의되는 기능 요건을 만족하는 화면, 장표 등 클라이언트상에서 동작하는 소프트웨어 구성 요소(프로그램)에서 생기는 층이다.

④ 데이타베이스 층 : 시스템 개념으로부터 정의된 정보요건을 만족하는 서버상의 데이터베이스스키마(데이타 모델)의 층이다.



(그림 2) 시스템의 구조 모델
(Fig. 2) Structure Model of System

3.2 소프트웨어 프로세스 구조 모델

소프트웨어 프로세스의 구조는 (그림 3)과 같이 모델화한다. 이 구조 모델은 시스템 층을 종축으로 채택하고 소프트웨어 라이프사이클 프로세스를 횡축으로 채택한 메트릭스 구조를 가진다. 여기서 소프트웨어의 라이프사이클 프로세스는 공통 프레임[4]으로 정의된 프로세스로 구성되어 있다. 이 소프트웨어 프로세스 구조 모델 상에서는 4가지 작업흐름의 프로세스가 정의된다.

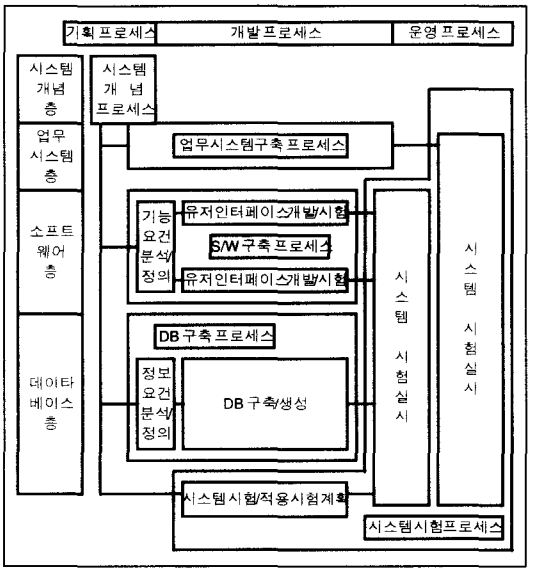
(1) 시스템 개념 구축 프로세스 : 경영 목표 실현을 위하여 시스템화의 목적, 목표 및 시스템화의 범위를 설정(목표설정)하고 시스템의 비용대 효과를 평가하기까지의 프로세스이다.

(2) 업무시스템 구축 프로세스 : 업무 설계로 시작, 전표, 장표의 설계, 업무 매뉴얼의 작성 등 업무기능관계(인간관계)의 시스템을 구축하는 프로세스이다.

(3) 데이타베이스 구축 프로세스 : 정보 요건의 분석, 정의로 시작, 데이타베이스 서버상에서 데이타베이스를 생성하기까지의 프로세스이다.

(4) 소프트웨어 구축 프로세스 : 기능 요건의 분석, 정의로 시작, 클라이언트에서 동작하는 화면과 장표의 프로그램을 생성, 시험, 검증하기까지의 프로세스이다.

(5) 시스템 시험 프로세스 : 시스템 시험 계획으로 시작, 시스템시험 실시에서 시스템 운용시험 실시까지의 프로세스이다.



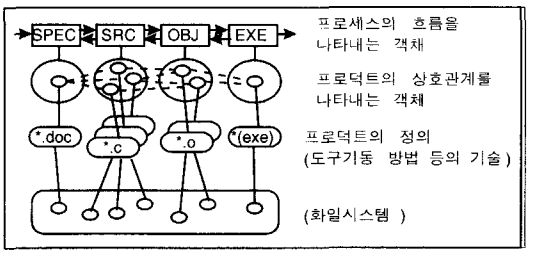
(그림 3) 소프트웨어 프로세스의 구조 모델
(Fig. 3) Structure Model of Software Process

3.3 프로세스 기술 모델

(1) 기술 모델

개발 프로세스를 기술하기 위해 객체에 기초한 제안 모델은 독립적으로 기술 가능한 세 개의 정의부분으로 되어 있으며 기술된 프로세스는 (그림 4)와 같은 구성을 가진다.

구체적인 프로세스의 흐름은 그림 상단의 객체간 조작의 흐름으로 표현된다. 각각의 객체는 개발 작업에 의해 작성되는 제품의 집합을 나타내고 있다. 이 부분에서는 제품의 집합에서의 조작과 집합간의 관계를 기술한다. 그림 가운데는 제품 집합의 요소와 그들간의 상호 관계를 나타내는 객체이다. 객체 집합에서의 조작이 개개의 제품에 어떻게 적용되는가 또한 개개의 제품간에 어떤 상호관계가 있는가를 기술하는 부분이다. 그림 맨 아래는 화일시스템이고 그 위에 속성, 조작을 가진 객체로서 개개의 제품이 기술된다. 각각의 제품에 대한 조작이 어떻게 실현되는가를 기술하는 부분이다.



(그림 4) 프로세스와 제품의 기술 모델

(Fig. 4) Dscription Model of Process and Product

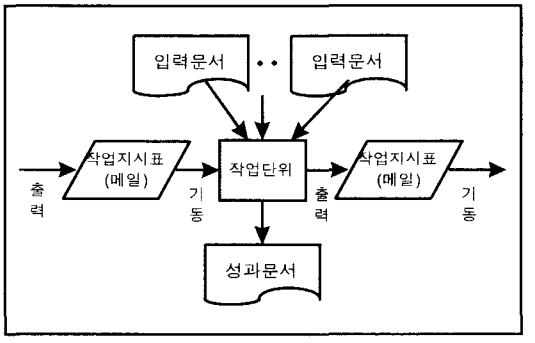
(2) 작업흐름 구조 모델

소프트웨어 프로세스에서 작업흐름란 소프트웨어 프로세스를 구성하는 작업, 작업 지시, 작업보고와 입력 및 출력 문서의 흐름이라고 정의할 수 있다. 즉, 설계 정보가 문서의 형태로 전달되고, 작업지시와 작업관리 정보가 작업관리표의 형태로 전달되는 일련의 작업 흐름이 작업흐름이다. 이 작업흐름 구조를 모델화한 것이 (그림 5)의 모델이며 작업흐름의 구조는 다음과 같이 기술된다.

첫째, 작업흐름에서 정보 공유는 문서를 개입시킴으로서 실현된다.

둘째, 작업단위의 기동은 지시표에 의해서만 발생한다.

셋째, 작업흐름 작업은 작업단위로 분해되고 작업단위로 역할이 할당된다.



(그림 5) 작업흐름의 구조 모델

(Fig. 5) Structure Model of Workflow

4. 작업흐름의 분석

작업흐름 분석의 목적은 현장의 소프트웨어 프로세스에서 객체가 실현하고 있는 작업흐름을 명확하게 역공학함으로써 작업흐름 설계에 필요한 작업흐름 모델을 구축한다. 작업흐름 분석에는 다음 2가지의 단계가 포함된다.

4.1 객체의 추출

소프트웨어 프로세스가 기술되어 있는 매뉴얼 종류를 작업흐름의 관점에서 분석하여 작업흐름을 구성하는 4가지의 요소(객체) 즉, ①문서, ②작업 단위, ③역할, ④작업관리표를 추출한다.

(1) 문서

작업흐름에서 설계 정보 등의 정보 공유는 명세서 등의 문서를 개입시켜 실현한다. 문서에는 ①기능요건 정의서, ②화면 설계서, ③데이타베이스 설계서, ④데이타베이스 설계심사통지서, ⑤목표설정서, ⑥화면 프로그램 등이 포함된다.

(2) 작업단위

작업단위는 소프트웨어 프로세스에서 작업의 최소 단위이며 그 성과물로서 하나 이상의 문서가 규정되어 있으며 공통 프레임에서는 활동 타스크(Activity Task)가 정의되어 있다. 통상 소프트웨어 프로세스에 대해서는 작업 공정표의 세부 항목이 작업 단위에 해당하는 것이 많다. 작업단위의 예로서 ①문제점 분석작업, ②기능요건 정의작업, ③정보요구 정의작업 등을 들 수 있다.

(3) 역할의 추출

역할은 주어진 임무에 따라 소프트웨어 프로세스에 종사하는 각 시스템 엔지니어 또는 소프트웨어 엔지니어를 나타내고 있다. 따라서 역할은 프로젝트 체제표를 분석하여 추출할 수 있다. 역할에는 ①업무 설계 담당, ②화면 설계 담당, ③데이타베이스 설계 담당, ④데이타베이스 설계 심사원, ⑤목표설정 승인자, ⑥화면 설계 승인자 등이 있다.

(4) 작업관리표의 추출

문서는 문서공유 기능은 가지고 있지만 작업흐름에 비해 능동적인 작업지시, 관리 등의 기능을 갖지 않는다. 이것에 비해 작업관리표는 소프트웨어 프로세스에 있어서 작업의 지시, 작업의 보고를 목적으로 한 전자화된 문서(전자 메일)로서 하드웨어의 생산관리 시스템에 있어서 작업 지시표, 작업보고에 대응하는 것이다. 작업흐름의 진척에 따라 작업단위의 성과인 문서를 작업관리표에 차례로 첨부시켜 행하게 된다. 작업관리표

의 예로는 ①업무설계 작업 지시표, ②화면 소프트웨어 개발작업 지시표, ③데이타베이스 구축 작업 지시표, ④개념 구축 작업 지시표, ⑤업무시스템 구축 작업 지시표가 있다.

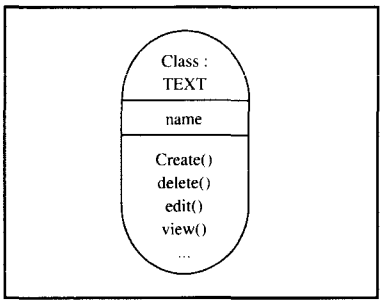
4.1.1 객체의 도입

개발 작업이 진행됨에 따라 생성되고 사용하는 여러 가지 제품을 기술하기 위해 객체의 개념을 도입한다. 개발 작업중에 사용되는 제품은 그 형식과 용도 및 내용에 따라 몇가지 종류로 나눌 수 있다. 제품은 이들 종류마다 공통으로 가지는 속성이나 조작에 대한 기술을 클래스로 정의한다. 실제 각각의 제품은 이 클래스의 인스턴스 객체로서 동적으로 생성되는 것이다.

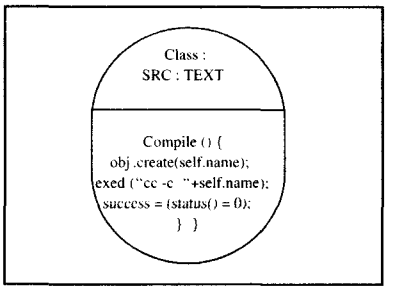
본 논문에서는 클래스 정의에 대해서 상세하게 기술하지는 않았지만 정수, 문자열 등의 데이타형 및 배열과 리스트 등의 데이타 구조는 이미 정의되어 있는 것으로서 기술중에 자유로이 이용하는 것으로 하였으며 클래스의 기술에는 절차 언어의 기술형식을 이용한다.

4.1.2 계승에 의한 클래스 정의

새로운 클래스를 정의할 때 그 정의가 이미 존재하는 클래스의 정의를 포함하고 있는 경우가 있다. 이 경우 이미 존재하고 있는 클래스 정의를 계승하여 새로운 클래스의 정의를 간결하게 할 수 있다. 예를 들면 텍스트 화일을 나타내는 클래스 TEXT를 미리 정의해 두고 외부에서 사용가능한 속성과 집합인 인터페이스를 (그림 6)과 같이 기술한다. 이 때 새로운 C의 소스 프로그램을 나타내는 클래스 SRC를 클래스 TEXT의 정의에 compile이라는 조작을 부가한 것으로 (그림 7)과 같이 정의할 수 있다. 즉, exec는 도구를 기동하기 위한 함수이며 self는 그 인스턴스 객체 자체를 나타내는



(그림 6) 클래스 TEXT의 인터페이스
(Fig. 6) Interface of Class Text



(그림 7) 클래스 SRC의 정의
(Fig. 7) Definition of Class SRC

기법으로 클래스의 계승 구조를 도입함으로써 여러 가지 제품의 클래스를 보다 간단하고 나누기 쉽게 기술할 수 있다.

4.1.3 객체의 병렬 실행

소프트웨어 개발 작업에서는 몇 개의 작업을 병렬로 실행하고 싶은 것이 있다. 예를 들면 요구명세서를 작성하면서 모듈 설계를 하거나 관련된 몇 개의 소스 프로그램을 별개의 윈도우에서 병행하여 편집하는 것이 빈번하게 발생한다. 위와 같이 각각의 제품이 하나의 객체로 표현되고 있는 경우 이것은 복수의 객체를 병렬로 실행하는 것에 해당한다.

복수 객체의 병렬 실행에 대해서는 여러 가지 방법이 제안되고 있으나 본 연구에서는 obj를 객체, op()를 조작으로 할 때 obj.op()에서 조작을 순차적으로 기동하는 것과 obj<-op()에서 조작을 현재의 작업과 병렬로 기동하는 것을 나타내는 것으로 한다[12]. 그리고 순차적인 기동의 경우 기동한 조작의 동작이 종료하기까지 다음의 동작으로 이동하지 않지만 병렬로 기동한 경우에는 기동한 작업과 현재의 작업을 병렬로 수행할 수 있다. 하나의 객체가 한번에 실행 가능한 조작은 하나뿐이라고 가정한다.

PDL에 의한 프로세스 기술의 경험에서 소프트웨어 개발의 경우 실제로 병렬로 수행하는 작업의 수는 많아야 4개 정도이다. 이것은 개발자가 다수의 작업을 동시에 처리하는 것이 가능하지 않기 때문이고 역으로 작업의 수를 지나치게 늘리면 작업을 수행하기 어려워지기 때문이다. 병렬로 실행하는 객체의 수를 필요이상으로 늘리지 않기 위해 현재 실행되고 있는 객체 수 등의 정보를 실행중에 동적으로 이용하여 객체의 동작을 제어하는 것을 생각할 수 있다.

4.2 작업흐름 모델의 구축

여기에서는 추출된 객체의 상호 관계를 명확히하여 작업흐름 모델을 구축하는 것으로서 여기서 작업흐름의 모델은 정적 모델과 동적 모델로 구성한다.

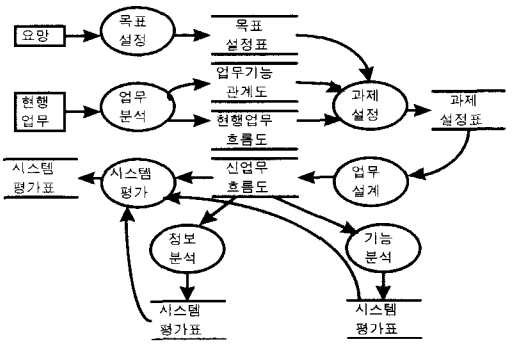
4.2.1 정적 모델

소프트웨어 프로세스를 구성하는 작업, 역할, 문서간의 정적인 상호관계를 표현하는 모델로서 DFD(Data Flow Diagram)에 의해 표현한다. 여기서 객체간의 상호관계는 다음과 같이 명확화된다.

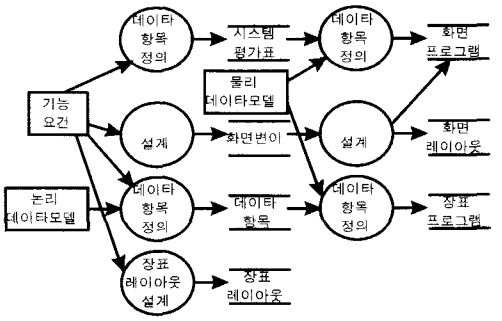
① 작업단위와 문서의 관계는 작업단위마다 성과 문서가 규정되어 있다.

② 그러나 작업단위로 참조되는 문서는 규정되어 있지 않은 것이 많다. 이 경우에는 시스템 개발의 매뉴얼로 되돌아갈 필요가 있다.

(그림 8)은 시스템 기획 프로세스와 소프트웨어 구축 프로세스의 정적 모델을 나타낸 것이다.



(a) 시스템 기획 프로세스



(b) 소프트웨어 구축 프로세스

(그림 8) 작업흐름의 정적 모델도

(Fig. 8) Static Model of Workflow

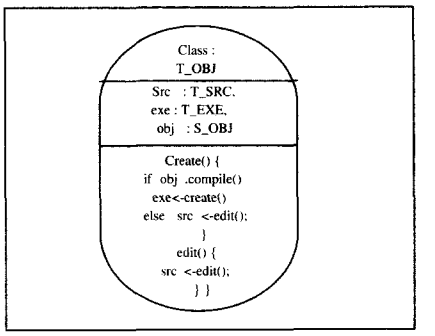
(1) 프로세스 흐름의 기술

프로세스 흐름은 개발중에 작성되는 여러 가지 제품을 나타내는 객체와 그 사이의 조작의 기동에 의해 표현하는 것이 가능하다. 예를 들면 C 프로그램을 개발하는 간단한 프로세스는 (그림 4)의 상단과 같은 제품 간의 관계로 표현할 수 있다. 우선 명세로부터 소스 프로그램에 대하여 프로그램 생성의 조작이 기동되고 소스 프로그램이 작성 가능하다면 객체 프로그램(OBJ)에 대하여 생성의 조작(컴파일)이 기동된다. 컴파일이 성공하면 실행 프로그램(EXE)에 대하여 생성 조작(링크)이 기동되지만 실패하면 소스 프로그램에 대하여 편집의 조작이 기동된다. 이와 같은 관계를 기술하기 위해 제품의 집합을 나타내는 객체는

- 입력 제품의 집합
- 출력 제품의 집합
- 조작(생성, 편집 등)대상이 되는 제품의 집합

을 나타내기 위한 속성과 조작의 정의를 가진다.

(그림 4) 가운데 객체프로그램 부분의 조작의 흐름을 클래스로 기술하면 (그림 9)와 같으며 조작의 대상은 다음 절에서 서술하는 상호 관계를 나타내는 객체의 예이다.



(그림 9) 프로세스 흐름의 정의 예

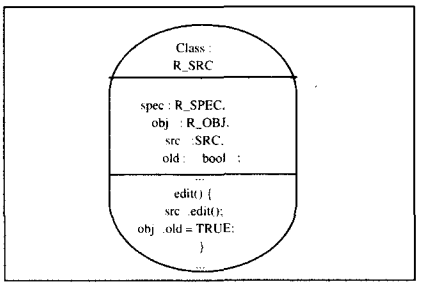
(Fig. 9) Example of Definition of Process Flow

4.2.5 동적 모델

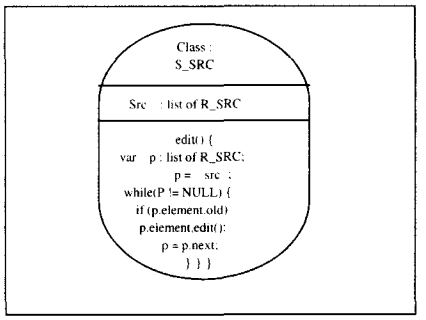
(1) 제품 상호관계의 기술

개발작업을 효율적으로 지원하기 위해서는 여러 가지 제품 간의 관계에 대한 정보를 이용할 필요가 있다. 구체적인 각각의 관계는 개발작업중에 동적으로 변화해 가기 때문에 미리 작성 도중에 기술하는 것은 불가능하지만 어떤 클래스의 제품이 어떤 종류의 상호관계를 가지는 것은 미리 기술할 수 있다. 또한 복수의 제품을

집합으로 취급하는 경우 각각의 제품을 구체적으로 몇 가지 생성하며 어떤 상호관계를 가지는가에 대해 개발 작업중에 동적으로 대응해야 한다. 제품 간의 상호관계와 상호 관계에 기초한 조작을 기술하는 객체 및 그것을 집합으로서 취급하기 위한 객체를 이용하여 복수 제품의 조작을 기술한다.



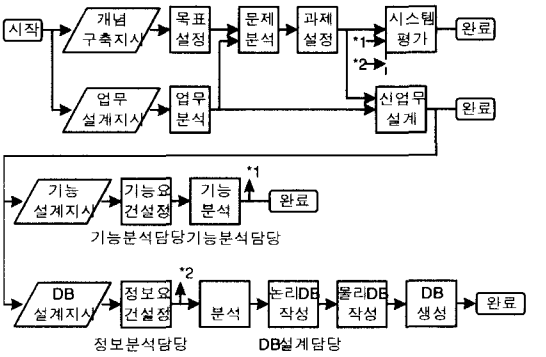
(그림 10) 상호 관계의 정의
(Fig. 10) Definition of Mutual Relation



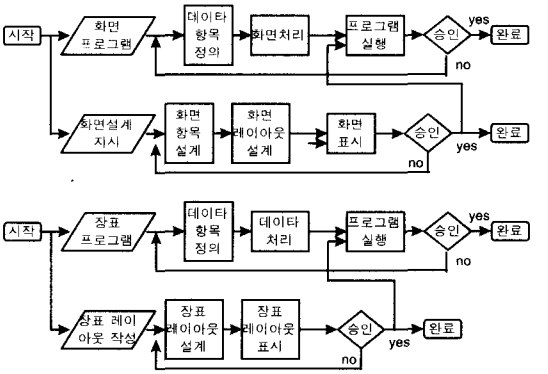
(그림 11) 상호 관계 집합의 정의
(Fig. 11) Definition of Mutual Relation Set

(그림 10)은 (그림 7)의 클래스 SRC의 상호관계를 정의하는 클래스 R_SRC이다. 소스 프로그램이 편집되면 그것으로부터 작성되는 객체 프로그램도 새롭게 작성해야 한다. 여기서 소스 프로그램의 편집을 수행한 경우 대응하는 객체 프로그램이 상호관계를 나타내는 객체의 old라고 하는 속성을 정확히 한다. 다음에 객체 프로그램을 작성할 때에 old가 정확히 되어 있는 제품에 대해서만 컴파일을 수행하면 좋다. (그림 11)은 R_SRC의 인스턴스 객체의 집합을 정의하는 클래스 S_SRC이다. 여기서는 edit라는 조작이 기동되었을때 리스트로 표현되고 있는 객체 가운데 old가 정확한 것에 대해서 차츰 edit 조작을 적용하는 것으로 기술하였다.

그리고 소프트웨어 프로세스의 시계열적인 움직임을 나타내는 모델로서 작업흐름 다이아그램(또는 작업흐름)으로 표현되며 프로세스마다 작업관리표를 마련하고 있다. 작업관리표는 프로세스의 개시시점(입구)에서 생성되고 그 프로세스 완료시점(출구)에서 소멸된다. 또한 프로세스 도중에 흐름의 분기가 발생하면 복제하고 흐름이 합류하는 시점에서 통합화된다.



(a) 시스템 기획 프로세스



(b) 소프트웨어 구축 프로세스

(그림 12) 작업흐름의 동적 모델도
(Fig. 12) Dynamic Model Chart of Workflow

(그림 12)는 (a) 시스템 기획 프로세스와 (b) 소프트웨어 구축 프로세스의 동적 모델을 표현한 것이다. 이 흐름도에 있어서 최상단은 시스템 개념 구축의 프로세스, 두 번째단은 업무시스템 구축 프로세스, 세 번째단은 소프트웨어 구축 프로세스이고 최하단은 데이타베

이스 구축 프로세스이다.

5. 작업흐름 설계 사례

5.1 프로세스 기술의 순서

본 장에서는 클라이언트/서버 시스템형의 하드웨어 네트워크 시스템에서 구축되는 소프트웨어 프로세스의 작업흐름 관리에서 프로세스의 기술을 다음과 같은 순서로 진행하였다.

첫째, 개발 작업 중에서 이용되는 제품을 열거하고 어떤 제품에서 어떤 제품이 작성되는가를 조사하며 그에 따라 프로세스의 흐름을 결정한다. 이 단계에서는 소스 프로그램 등 몇 개의 제품 집합은 하나의 통합으로 본다.

둘째, 제품의 집합에 대한 조작이 각각의 제품에 어떻게 적용되는가를 결정한다.

셋째, 각각의 제품이 어떤 상호관계를 가지고 있는가를 조사한다. 또한 속성, 조작과의 관계도 조사한다.

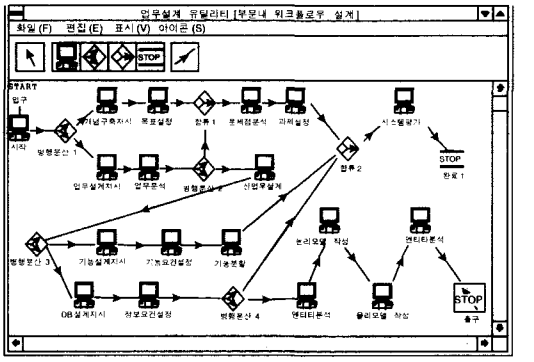
넷째, 제품의 속성과 조작을 어떠한 화일과 도구 등으로 실현하는가를 결정한다.

다섯째, 프로세스의 흐름과 제품의 상호관계 및 제품의 조작에 대하여 클래스를 정의한다.

작업흐름의 설계는 작업흐름 분석으로 구축된 작업흐름 모델을 작업흐름 소프트웨어의 설계명세서로서 정의하는 것이다. 여기에서는 명세정의 대상의 개발중인 작업흐름설계소프트웨어인 WORKMAN(WORKflow MANagement system)을 이용하였다. 작업흐름 설계의 순서는, ① 플로우 설계 ② 작업단위 설계 ③ 작업관리표 설계의 세 단계로 구성된다.

5.2 플로우 설계

플로우 설계는 작업흐름 분석된 동적 작업흐름 모델을 작업흐름 소프트웨어로 명세정의화하는 것이다. WORKMAN에서 플로우 설계에서는 업무 설계 유틸리티의 GUI를 사용한 명세정의가 가능하다. (그림 13)은 (그림 12) (a)의 기획 프로세스에 대응한 작업흐름 명세 정의로서 즉, (그림 8)의 상부에 기술되어 있는 것처럼 작업흐름의 ① 입구, ② 작업단위(Stage), ③ 분기, ④ 합류, ⑤ 출구 등에 대응한 아이콘이 준비되어 있고, 이들 아이콘과 아이콘 사이를 접속함으로서 명세정의가 실현된다. 플로우 중의 분기에 대해서는 분기조건을 조건식의 형식으로 정의할 수 있다.



(그림 13) 기획 프로세스의 작업흐름 정의 화면의 예
(Fig. 13) an Example of Work-flow Definition Screen of Project Process

5.3 작업단위의 설계

여기에서는 작업흐름 분기에서 DFD형식으로 모델화시킨 작업 단위와 도큐멘트의 상호 관계를 작업 단위(Stage) 정의(WORKMAN에서는 작업 단위를 Stage로 취급)로써 명세 정의한다. 즉, ① 작업 단위(Stage명), ② 대상 도큐멘트명(구성 파일명), ③ 담당 역할(Role명), ④ 처리 기한 등을 정의한다.

5. 분석 및 고찰

5.1 형식적인 의미부여

개발 프로세스의 기술은 처음에 추상도가 높은 기술을 적용하고 이것을 순차적으로 상세화하는 방법으로 작성하는 경우가 많다. 상세화의 각 단계에서는 새롭게 얻은 기술이 전 단계의 기술을 바르게 상세화해 가는가의 여부를 검증할 필요가 있다. 이 때 여러 가지 상세화 레벨의 기술이 몇 가지 동일한 형식적인 구조 중에서 의미가 부여되었다고 하는 검증을 하는 것이 가능하다.

본 연구를 통하여 프로세스 기술 언어 PDL을 대표적 명세 기술언어 ASL[6]의 의미부여에 기초한 함수형 언어로 설계하였다. ASL은 동일한 형식적인 의미의 구조가운데 여러 가지 추상화 레벨의 기술 및 상세화를 간결하게 행할 수 있다. 또한, PDL은 이 ASL의 성질을 계승하고 개발 프로세스의 기술을 단계적 상세화에 의해 쉽게 작성할 수 있다. 클래스의 기술에도 마찬가

적로 의미부여를 도입함으로써 단계적 상세화에 의한 기술의 작성과 검증을 쉽게 할 수 있다고 기대된다. 또한 클래스 정의를 부품으로서 재이용하는 경우 정확성의 검증 등을 행하는 것도 생각할 수 있다. 그러나 ASL은 절차 언어 의미에서의 변수를 갖고 있지 않기 때문에 위에서 서술한 바와 같이 클래스 정의에 그대로 대수적 명세기술의 의미부여를 하는 것은 가능하지 않다. 또한 계승을 하는 클래스 정의의 의미부여에 대해서도 검토할 필요가 있다.

5.2 제품 서버의 도입

본 연구의 제안 모델에서는 제품에 대하여 구체적인 화일 조작을 행하는 부분은 클래스 내부에 정의로 기술하는 것으로 되어 있다. 프로세스 기술의 실행 시스템이 화일 조작에 대해서 어떤 지원도 하지 않는 경우 이들 조작의 정의는 OS가 제공하는 기능을 직접 이용하여 기술해야 한다.

이에 비해 제품에 대한 구체적인 조작에 대해서는 시스템이 어느 정도의 기능을 제공하고 기술은 이들 기능을 편성하는 것만으로 정의할 수 있도록 하는 방법을 생각할 수 있다. 시스템 내에서의 제품의 조작관리를 행하는 부분을 제품 서버라고 부른다. 제품 서버의 사고방식을 도입하는 것에 의해 제품의 조작을 쉽게 기술할 수 있다고 기대한다. 그러나 제품 서버 자체의 필요성도 포함하고 어떠한 레벨에 어떤 기능을 제공해야 하는가에 대해 충분한 검토가 이루어지지 않고 있다. 향후 여러 가지 프로세스를 기술하여 경험을 축적할 필요가 있다.

5.3 작업흐름 관리효과

소프트웨어 프로세스에서 작업흐름 관리의 첫번째 효과는 문서 루틴의 리드타임(Read Time) 해소와 작업지시 및 작업 대체시간의 단축이다. 두번째는 프로세스 진척관리 정밀도의 향상이다. 즉, 작업흐름 관리에 의해 고정 지연의 문제발생에서 대처하는데까지의 관리 프로세스의 사이클이 단축되는 것이다.

작업흐름 관리에서 이들 효과를 정량적으로 평가하기 위해서는 현행의 작업흐름 관리가 되고 있지 않은 소프트웨어 프로세스에 대해 작업흐름 관리를 도입한 경우의 프로세스를 시뮬레이션에 의해 평가할 수 있다. 우선, 현행의 소프트웨어 구축 프로세스에서 진척 실적을 측정한 다음 작업흐름 관리를 도입한 경우의 진척을 시뮬레이션에 의해 평가하여야 한다. 이 소프트웨어 구축 프로세스 사례에서 작업흐름 관리도입의 효과를 소프트웨어 메트릭스로 평가하면 리드타임이 10일간 단축되었으며 소프트웨어의 생산성은 약 15% 향상되었다는 것을 알 수 있다[13].

5.4 팀 개발에 의한 확장

본 논문에서는 프로세스를 한 사람의 개발자가 행하는 작업으로 한정하여 생각하였다. 그러나 각 객체의 병렬 동작을 협조하여 일하는 팀 내부의 개발자라고 보지 않으면 본 논문의 모델을 팀에 의한 개발에 확장하는 것이 가능하다고 생각한다.

이 경우 각 개발자간의 메시지 교환과 분산 자원 및 제품의 관리를 어떻게 하는 것이 좋은가라는 것이 문제가 된다. 또한, 각 작업자를 어떤 작업에 배치할 것인가, 전체 작업의 관리·운영은 어떻게 진행되는가 등이라는 인적 측면에 대한 분석과 연구가 필요하다.

6. 결 론

소프트웨어 프로세스 개선 방법을 이용하여 개발 프로세스를 향상시키는 방법에 관한 것으로서 대규모 소프트웨어 개발 프로세스의 작업흐름을 분석하고 품질을 개선시키는 활동에 관한 내용이다. 본 연구의 목적은 소프트웨어 프로세스의 작업흐름 관리 시스템 구축을 위한 워크 플로우 분석과 구축을 위한 것으로서 그 성과는 다음과 같다.

첫째, 프로세스의 기술과 산출물 관리의 중요성을 인식하고 소프트웨어 프로세스의 구조를 모델화하고,

둘째, 소프트웨어 프로세스의 구조를 분석함으로써 소프트웨어 프로세스의 작업흐름 모델을 구축하는 작업흐름 분석 방법을 정립하였으며,

셋째, 작업흐름 소프트웨어 WARKMAN에 의해 명세성의 기법의 설계 사례를 기술하고 연구 결과의 분석과 고찰을 하였다.

즉, 본 논문에서는 객체지향에 기초한 프로세스 기술 모델을 제안하여 개발 프로세스를 제품 간 상호관계와 조작의 기동에 의해 기술하였다. 이와 같이 함으로써 병렬의 복수 개발작업을 객체의 병렬동작으로 기술하는 것이 가능하고 또한 실행중에 객체를 동적으로 결합시키는 방법에 의해 각 개발작업에 적

합한 지원이 가능하다.

향후 과제로서 도중에 프로세스 또는 작업흐름 변경이 발생한 경우의 처리방법과 정형적인 프로세스뿐만 아니라 명세 변경관리 프로세스 등의 비정형적인 프로세스의 취급에 있어서 작업흐름 관리 방법 및 소프트웨어 프로세스에서 공정, 품질, 원가 관리 데이타의 자동 수집방법에 관한 연구이다.

참 고 문 헌

[1] Aoyama, M., "Concurrent Development Process Model", IEEE Software, Vol. 10, No. 4, pp. 46-55, 1993.

[2] M. Hammer et. al., "Reengineering the corporation - a manifest for business revolution, M. Hammer and James Company", NY, 1993.

[3] Mi.P., Scacchi,W.,"Process Integration in CASE Environments", IEEE Software, Vol. 9, No. 2, pp. 45-53, 1992.

[4] O. J. Dahl and K. Nygaard, "Class and subclass declarations", IFIP Working conference on Simmulation Programming Languages, North-Hooland, 1968.

[5] K. Inoue, T. Ogihara, T. Kikuno and K. Toril, "A Formal Adaptation Method for Process Descriptions", Proc. of 11th Intenational Conference on Software Engineering, pp. 145-153, 1989.

[6] Perry, D. E. et al., "People, Organizations, and Process Improvement", IEEE Software, Vol. 11, No. 4, pp. 36-45, 1994.

[7] 村上, ソフトウエアライフサイクルプロセス, 情報處理, Vol. 36, No. 5, 1995.

[8] 石田, ソフトウエア開發プロセスの再構築, 情報處理, Vol. 36, No. 5, 1995.

[9] 荻原, 井上, 鳥居, "ソフトウェア開發を支援するツール起動自動制御システム", 電子情報通信學會論文誌(D-I), J72-D-I, 10, pp. 742-749, 1989.

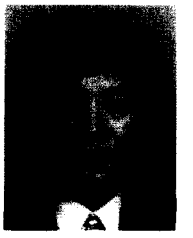
[10] 김소연, 이강수, "워크플로우 모형화 및 관리시스템", 한국정보처리학회, 「정보처리」Vol. 2, No. 3, 1995. 9.

[11] 양해술, 이용근, 황인수, "소프트웨어 프로세스 리엔지니어링(SPR)의 개요와 접근 방법", 한국정보처리학회, 「정보처리」 Vol. 2, No. 3, 1995. 9.

[12] 양해술, 이용근, "소프트웨어 프로세스의 워크플로우 분석과 품질개선 활동," 한국정보처리학회, 「정보처리」 Vol. 3, No. 5, 1996. 9.

[13] 양해술, "소프트웨어 워크플로우 분석과 설계 방안," 한국정보처리학회, 추계학술발표논문집, 1996. 10.

[14] 양해술, "한진해운 신정보(영업 및 물류)시스템의 품질감리와 평가", 한진해운, 평가보고서, 1998. 2.



양 해 술

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)

1878년 성균관대학교 정보처리학과(석사)

1991년 日本 오사카대학교 기초공학부 정보공학과 S/W 공학 전공(공학박사)

1975년~79년 육군중앙경리단 시스템분석장교

1986년~87년 日本 오사카대학교 객원연구원

1980년~95년 강원대학교 전자계산학과 교수

1993년~94년 한국정보과학회 학회지 편집부위원장

1994년~95년 한국정보처리학회 논문지편집위원장

1994년~현재 한국산업표준원(KISI) 이사

1995년~현재 한국소프트웨어품질연구소(INSQ) 소장

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