□특집□

소프트웨어 프로세스 자동화

안 유 환† 이 양 규†† 이 승 지††† 김 길 조††††

◈ 목 차 ◈

1. 서 론

소프트웨어가 복잡해지고 규모가 대형화하면서 소프트웨어 생산을 위한 여러 팀의 공동 작업 요구가 커지고 있다. 이러한 공동작업은 구조적이고 훈련된 환경에서 조정·통제될 필요가 있기 때문에 효율적이고 효과적인 소프트웨어 프로세스 관리가 점점 중요한 요인이 되고 있다[1]. 6~12개월의 개발노력이 필요한 프로젝트에 대한 방법론은 종종 400~500개의 식별 가능한 작업(task)을 가지기 때문에 이러한 많은 수의 작업들을 수행하고 관리하는 일 때문에 생산성 및 품질향상의 문제는 통제 불가능하게 된다[7]. 이 때문에 Watts Humphrey는 원하는 목적을 달성하기 위하여 좋은 프로세스 관리를 채택하는 것 외에도 보다 진보된 도구, 기술 및 관리를 필요로 한다고 하였다.

† 정회원 : 시스템공학연구소 선임연구원
†† 정회원 : 서원대학교 경영정보학과 교수
††† 정회원 : 시스템공학연구소 선임연구원
†††† 정회원 : 시스템공학연구소 연구원

즉, 자동화의 도움 없이 프로세스 관리 원칙을 적용하는 것도 프로세스 개선을 위한 기본적인 기반이 되나, 자동화된 프로세스 관리 시스템을 포함하는 애플리케이션 개발 솔루션은 애플리케이션 개발노력의 관리 및 지속적 개선을 위한 보다 일관되고 반복 가능한 기반을 제공한다는 것이다.

소프트웨어 프로세스란 소프트웨어 개발, 운용, 보수에 관한 작업과 그 연결 및 그것들에 영향을 미치는 각종 요인 (사람, 기술)으로 정의된다. 최근에는 생산성 향상과 품질 개선을 위하여 제품 및 서비스를 개발·배치·유지 보수하는 프로세스에 관심의 초점이 맞춰지고 있다. 소프트웨어 공학 분야에서도 대규모의 다양한 조직에서 사용되는 프로세스 기술 및 관습들을 분석하여 개선된 프로세스를 개발하려는 노력이 계속되고 있다. 이러한 것은 CMM (Capability Maturity Model)과 같은 프로세스 개선 전략으로부터 소프트웨어공학 프로세스를 표현하는 모델링 기법, 소프트웨어 프로세스를 실행하기 위한 도구 및 환경, 프로세스 데이터를 수집·분석하기 위한 기법 등을 포함하고 있다[2].

본 고에서는 소프트웨어의 품질과 생산성 향상, 지속적인 프로세스의 개선에 필수적인 소프트웨어 프로세스 자동화의 정의, 배경, 향후 발전방향 등을 알아보고 이에 필수적인 기술인 모형화 기술, 프로세스 중심 통합환경 기술을 개괄적으로 고찰한다.

2. S/W 프로세스 자동화 개요

2.1 S/W 프로세스 관리 및 자동화의 정의

소프트웨어 프로세스 관리는 여러 가지 의미를 가질 수 있다. 넓은 의미로는 프로젝트 관리, 메트릭스, 품질관리 및 기타를 포함하여, 프로젝트 전체에 걸친 개발 팀의 모든 작업을 포함한다. 이러한 넓은 의미는 프로세스 성숙도의 의미와 ISO 9000 규격의 의미를 잘 반영한다. 종래에는 프로세스 관리를 프로젝트 내에서의 산출물(예: DFD, 모듈) 과 그 상태(예: 진행중, 완료) 및 관련된 활동들로 주로 언급하였으나 이러한 좁은 관점은 관리 측면을 무시하였으므로 최근에는 통용되지 못하는 개념이다.

이에 반하여, 프로젝트 관리는 전통적인 관점에서 프로젝트의 상세와 제약사항(누가 어떤 일을 언제)을 관리하는 것이다. 프로젝트 관리 기법의 사용으로 자원 및 노력의 산정, 일정계획, 품질목표 등의 작업계획이 설정된다. 계획, 추적, 측정, 기록이 프로젝트 관리 측면이다.

이를 자세히 보면, 프로세스 정의 측면에서 통합된 개발환경이 제공되지 못하던 때에는 넓은 의미의 프로세스 관리가 효과적으로 이루어질 수 없어 프로젝트 관리와 프로세스 관리 측면을 구분하고 있었다고 할 수 있다.

일반적으로 프로세스 자동화는 다음에 대한 관리를 도와주는 기술로 정의된다.

- o 프로젝트를 구성하는 활동들의 순서 안내
- o 개발되고 있는 산출물에 대한 관리
- o 개발자들의 작업 수행에 소요되는 도구의 Invoke
- o 사람의 작용을 필요로 하지 않는 자동적인 작업의 실행
- o 산출물을 구축하는 사람들간의 의사 소통
- o 메트릭 데이터의 자동화된 수집
- o 개인 업무관리의 지원
- o 사람에 의한 에러의 축소
- o 현행성 있고 정확한 상태정보에 의한 프로젝트 관리 제공

2.2 S/W 프로세스 자동화의 배경

기계나 전자제품 등의 제조 분야에서는 프로세스 자동화가 생산성을 높이는데 결정적인 영향을 미쳐왔으나, 소프트웨어에 대해서는 아직 일반적으로 사용되지 못하고 있는 기술이다. 일반 제조업 분야에서의 이러한 개념을 소프트웨어 제조 관점에서 보면 일부 특정 해결책의 차이를 제외하고는 인간 및 조직에 관한 문제에 있어서는 거의 동일하다고 볼 수 있다. 따라서, 공장이라는 개념을 소프트웨어 개발분야에 끌어들이기 위하여 미국과 일본의 회사를 중심으로 다양한 노력이 있어왔다. 일본에서는 NEC나 히다찌에서, 미국에서는 GTE, IBM이 선도적 역할을 하였으며, 이러한 주도는 소프트웨어를 개발하는 방법을 표준화하여 비용을 줄이고 품질을 높이기 위한 동기에서 시작되었다.

소프트웨어 공장은, 다양한 기술을 통하여 소프트웨어 생산을 합리화 하고자 시도한 것으로, 프로세스에 대해 수동적 구현 또는 부분적 자동화를 통하여 표준화를 시도하였다. 이들 프로세스 및 표준들을 분석하고 개선하기 위한 메트릭 데이터도 수집되었다. 또한, 프로젝트에 대한 기술

지원 제공, 소프트웨어 설계, 코드 및 문서의 재사용 등이 권장되었고, CASE도구가 적용되었다.

이러한 노력들은 1960년대 후반에서 시작되어 지금도 계속되고 있다. 미국에서는 IBM에서의 경험으로부터 진화한 CMM이 개발되었는데 이는 소프트웨어 조직의 성능을 개선하는데 현저한 영향을 주었다. 일본에서는, 히다찌를 중심으로 소프트웨어 개발에 대한 공장 개념의 연구를 계속하고 있으나, 일본 대기업의 소프트웨어 관련 조직에서의 효과성은 다른 분야의 효과성보다 아직도 뒤지고 있다.

소프트웨어 프로세스 자동화는 소프트웨어 공장이 추구하는 바 이상의 것이다. 프로세스 자동화는 개인용 컴퓨터와 워크스테이션의 광범위한 사용과 네트워킹 능력 증가의 결과에 따라 최근에야 현실적 의미를 갖게 되었다.

2.3 프로세스 개선과 프로세스 자동화

소프트웨어 산업이 발달하고 여러 가지 노력이 이루어지고 있음에도 불구하고 아직도 소프트웨어 산업은 여러 부문에서 Craft를 겨우 탈피하는 수준이다. 즉, 이제야 프로세스 개선을 위한 CMM을 적용한다든지, ISO 9000-3의 구현을 통해 변화하고자 하고 있다.

프로세스 자동화는 그 자신이 끝이 아니고 끊임없는 프로세스 개선을 위한 계속적인 노력의 일부이다. 이러한 노력은 많은 요소를 요구하지만 특히 중요한 것은 프로세스 자체를 이해하는 것으로서, 이해를 통해서만이 개선이 가능하다. 그러므로 현재의 프로세스가 실제로 어떻게 수행되는지를 이해하는 것이 첫 번째 단계이다. 프로세스는 몇 가지의 기술 중 하나에 의하여 표현될 수 있는데 이러한 모형은 실제 프로세스에 대한 합의, 프로세스의 추론, 잠재적 개선점의 식별을 위해서는 필수 불가결한 요소이다. 만일 프로세스 메트릭이 실제 운영 중에 수집된다면, 예를 들어 어디에서 병목현상이 발생하는지 보여줄 수 있어, 이러한 모형에 부가적 힘이 더해질 수 있다.

또한, 새로운 또는 개선된 프로세스를 설계함에 있어 이러한 프로세스를 모의실험하는 능력은 매우 가치 있는 일이다. 이러한 모의실험은 위에서 언급한 프로세스 모형에 기초하여 후보 프로세스에 대한 효과성 평가, 이들 행위에 대한 심사, 제안된 수정사항을 가정한 질문이 가능하도록 한다.

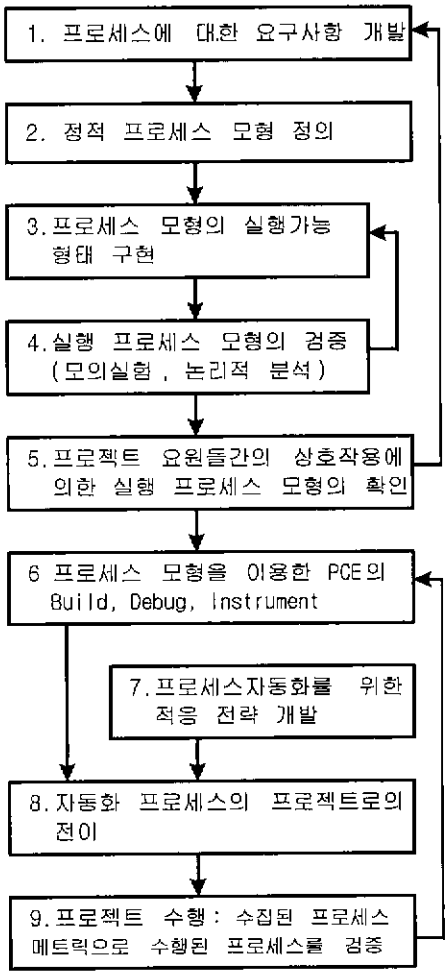
프로세스 모형은 나아가 수행 프로세스에 대한 검증을 지원하는데 이용될 수 있다. 높은 통합성을 요구하는 소프트웨어 응용환경에서는 정의된 프로세스 준수여부에 대한 보증은 기본적이다. 만일 적절한 데이터가 수집되면(예를 들면 수용된 의사결정, 이용된 제품의 버전 등) 프로세스 모형을 역추적 한다든지, 프로세스가 실제로 정확하게 지켜졌는지에 대한 검증이 가능하다.

2.4 프로세스 사용 시나리오

(그림 2.1)은 프로세스를 개발하고 이를 사용하여 프로세스 중심 통합환경(PCE: Process Centered Environment)을 구축하는 단계를 보여주고 있다. 단계1은 요구정의와 연계되고 단계 2부터 5까지는 프로세스 명세화 및 확인, 단계 6은 코딩과 테스팅, 단계 7부터 9까지는 프로세스 전개를 지원한다.

3. S/W 프로세스 모형화 기술

소프트웨어 프로세스 모형화는 소프트웨어 프로세스 모형을 개발하여 프로세스에 대한 정보 추출, 프로세스 평가, 대안 프로세스 중에서 최선안 선정, 새로운 프로세스의 설계, 기존 프로세스의 개선, 프로세스의 조정 및 재사용, 프로세스의 실행 등을 위한 기반을 제공한다.



(그림 2.1) 프로세스 사용 시나리오

프로세스 모형은 프로세스 중심 통합환경을 구축하기 위해 필요한 정보들을 많이 갖고 있기 때문에 프로세스 자동화를 위한 기초로서 사용될 수 있으나, 현재는 실시간 실행을 지원할 수 있도록 일반적으로 설계된 것은 아니다. 반면에 프로세스 중심 통합환경을 구축하기 위한 상업적 제품들은 프로세스 정의 및 모의실험을 위한 Graphical Front-ends를 제공하기 시작했다. 그러므로 앞으로는 프로세스 정의 모듈, 프로세스 모의실험 모듈 또는 실시간 프로세스 실행 모듈이나 이들이 호환적으로 조합된 형태로서 구입 가능한 모듈화된 제품을 볼 수 있다.

3.1 소프트웨어 프로세스의 특징

소프트웨어 개발 프로세스의 특징은 다음과 같다.

(1) 동시발생성 및 분산성

소프트웨어 프로세스는 시간적/공간적으로 분산되어 수행되는 연속적이고 병렬적인 작업들로 구성된다. 어떤 작업들은 여러 사람으로 구성된 팀에 의하여 컴퓨터의 네트워크 상에서 개발되기도 하고, 상호 협동적으로 수행되는 작업들이 있다.

(2) 불확실성 및 비결정성

소프트웨어 프로세스는 수행되는 세부 절차가 사전에 완전하게 예측되지 못하고 수행 과정에서 대안 선택이 일어나는 경우가 많다. 예를 들면 어떤 작업은 컴파일 결과에 따라 수행 내용이 달라진다. 반면에 선택해야 하는 내용이 설계에 대한 재검토가 수행되어야 한다던가 프로토타입이 개발되어야 한다는 등과 같은 사람이 관여하는 중요한 의사결정일 수도 있다.

(3) 오류, 재작업 및 기타 상황 의존성

소프트웨어 개발은 생성, 검사, 시험, 개선의 패턴을 따라 이루어진다. 오류는 전 단계의 작업까지 추적하여 전 단계의 작업을 다시 하거나 수정하여야 하는 경우가 종종 있다. 프로세스 모형을 개발할 때에도 추적, 작업 취소, 수정, 회복, 재세팅 등과 같은 오류를 처리하고 회복 방법이 포함될 수 있도록 하여야 한다.

(4) 진화 및 변경

소프트웨어 프로세스는 요구분석, 설계, 코딩, 시험, 유지 보수와 같은 생명주기를 갖는다. 생명주기 동안 소프트웨어 프로세스는 여러 가지 이유로 변경된다. 새로운 기술, 방법, 도구가 도입되

고 이에 따라서 프로세스도 재설계 된다.

3.2 S/W 프로세스 모형화의 목적

프로세스 모형을 구축하는 목적은 아래와 같다.

(1) 프로세스 이해

소프트웨어 개발에서도 개발자가 자신이 수행할 프로세스에 대하여 명확하게 알고 있지 못한다면 효과적인 작업성과를 기대할 수 없다. 프로세스 모형화는 프로세스에 대한 공통된 관점을 제공하기 때문에 개발자뿐만 아니라 사용자 등 관련 조직들에게 프로젝트의 프로세스에 대한 이해를 명확하게 해 줄 수 있게 된다.

(2) 프로세스 개선

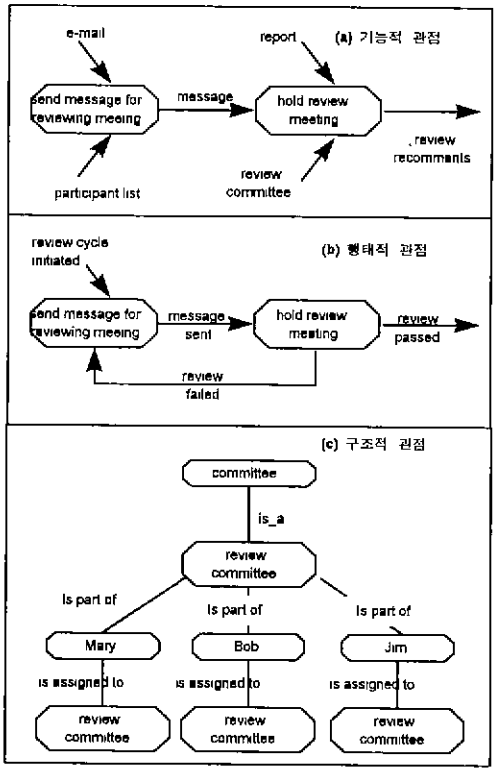
체계적인 프로세스 개선은 프로세스를 실행하여 얻은 경험이나 교훈을 새로운 프로젝트에 적용하여 점진적이고 지속적인 변화를 실현하려는 것이다. 이러한 개선은 SEI의 CMM 등과 같이 조직 내부의 목표 및 우선 순위 설정에 의하여 추진될 수도 있고 프로세스 모형화를 중심으로 이루어질 수도 있다. 프로세스 개선은 다음과 같이 이루어진다. 먼저, 현재의 상태가 파악되고 개선 목표가 수립된다. 현재의 상태를 모형화한 프로세스 모형이 구축되고 주요 성과 데이터들이 측정된다. 개선을 위한 변경들이 제안되고 변경이 포함된 새로운 프로세스 모형들이 실행되거나 모의실험된다. 변경된 모형의 데이터들을 분석하여 개선이 이루어졌는지를 평가한다. 개선이 이루어진 변경을 프로세스에 포함시킨다. 이러한 프로세스 개선을 위한 분석 데이터는 프로세스 모형을 이용하여 생성된다.

(3) 프로세스 실행

프로세스 실행은 프로세스의 수행이나 성과 등을 말한다. 프로세스의 실행은 안내(guidance), 자동화, 시행(enforcement) 등과 같은 수단을 통하여 이루어진다. 그 밖의 실행 활동으로는 추적, 감시, 메트릭 데이터 수집 등이 있다. 프로세스 모형은 몇 가지의 방법을 이용하여 실행을 시키고 결과를 파악할 수 있는 기능을 가질 수 있다.

3.3 S/W 프로세스 모형화의 관점

여러 가지의 프로세스 모형화 방법들이 있는데 각각의 방법들은 다른 관점을 강조하고 있으나 근본적으로 다음과 같은 세 가지로 구분할 수 있으며((그림 3.1) 참조), 이러한 관점의 혼합 형태도 가능하다. 실제 프로세스 모형화 방법에서는 혼합된 형태를 많이 사용한다.



(그림 3.1) 세 가지의 모형화 관점의 예

(1) 기능적 관점 (Functional View)

기능적 관점은 데이터 등과 같은 개체와 여기에 수행되는 기능에 초점을 둔 모형화 관점이다.

이러한 관점의 전형적인 예로는 구조적 분석에서의 자료흐름도를 들 수 있다.

(2) 행태적 관점 (Behavioral View)

행태적 관점은 활동이 수행되는 때와 조건을 강조하는 관점이다. 이러한 모형은 상태전이도에 근거를 두고 있다. 프로세스가 특정한 상태에 도달하면 어떤 사건이 발생하여 프로세스를 다른 상태로 전이시킬 수 있도록 한다.

(3) 구조적 관점 (Structural View)

구조적 관점은 다루어지고 있거나 프로세스에 의하여 변화되는 개체들간의 관계를 표현하기 위한 관점이다. 이러한 관점은 개체-관계도(Entity-Relation Diagram)에 근거를 두고 있다.

3.4 S/W 프로세스 정의 언어/시스템

(1) 비실행 패러다임

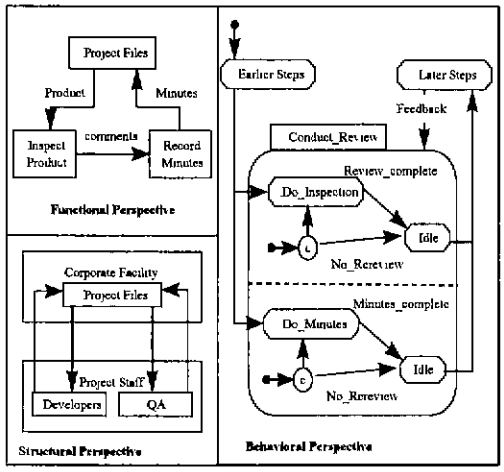
비실행 패러다임은 실행하는 어의(semantics)를 갖지 않는 텍스트 형태 및 그래픽 형태의 방법이다. 이것은 추상화된 프로세스 기술을 위하여 사용되고 주요 목적은 프로세스를 이해하는데 도움을 주기 위한 것이다. 이러한 방법의 예로는 IDEF0, IBM의 ETVX 등이 있다.

(2) 상태-기반 패러다임

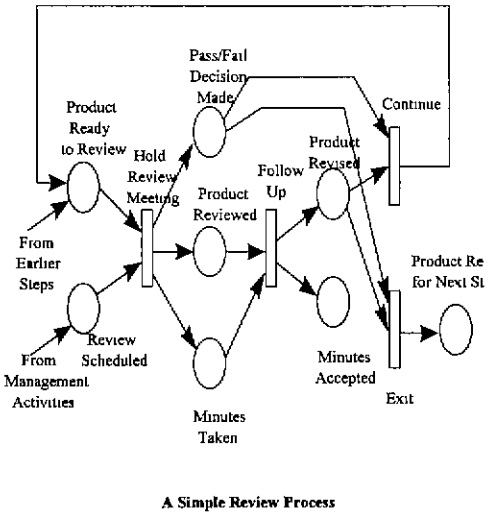
상태-기반 패러다임에는 크게 State-machine과 페트리 네트(Petri Net)가 있다.

Statemachine은 finite-state machine을 확장하여 계층적 구조를 표현한 것이다. Statemachine은 세 가지의 그래픽 언어를 이용하여 시스템의 구조적, 기능적, 행태적 관점을 표현한다. (그림 3.2)에 검토 프로세스에 대한 Statemate 모형화 결과가 나타나 있다.

또 하나의 상태-기반 방법인 페트리 네트는 (그림 3.3)과 같이 두 가지 종류의 노드(place, transition)와 이 노드들을 연결하는 화살 등을 이용하여 시스템의 동적인 형태를 표현한다.



(그림 3.2) Statemachine에서의 검토 프로세스



(그림 3.3) PetriNet에서의 검토 프로세스

페트리 네트와 페트리 네트의 확장된 형태를 이용하여 소프트웨어 프로세스를 모형화한 연구에는 MELMAC, SPACE 시스템의 SLANG, ProcessWEAVER 시스템 등이 있다[5].

(3) 규칙-기반 패러다임

규칙기반 시스템은 전문가 시스템, 프롤로그 등의 추론 기능을 이용하여 프로세스를 모형화 하

는 것이다. 이러한 방법의 중심 개념은 logic predicate의 데이터베이스이다. 이 predicate는 프로세스 산출물 모형(artifact model)을 나타내며 산출물의 동적 상태, 상호 관계, 속성을 표현한다. 규칙은 프로세스 단계를 의미하고 규칙들간에는 산출물의 상태를 통하여 상호 정보가 교환된다. 규칙 기반 시스템의 실행 규칙은 추론 기관의 행태에 의하여 결정된다.

MARVEL 시스템은 프로세스 실행을 지원하고 전문가 시스템을 기반으로 구축되었다. MARVEL의 법칙은 사전조건, 행동, 복수의 사후조건의 세 부분으로 구성되어 있다. 행동이 실행되기 위해서는 사전조건이 만족되어야 하고, 행동이 실행되면 사후조건에 따라 상태가 변경된다. 복수의 사후조건은 행동의 결과가 몇 가지로 나타날 수 있음을 의미한다. (그림 3.4)에 MARVEL의 예가 제시되어 있다.

(4) Imperative 패러다임

APPL/A는 Ada 언어 기반의 프로세스 모형화 언어이다. 이것은 범용 프로그래밍 언어이기 때문에 프로세스 모형뿐만 아니라 프로세스 엔진의 행태도 프로그램할 수 있다. 예를 들면 프로세스의 상태를 표현하고 프로세스의 이력을 기록하는 것이 APPL/A에서는 가능하다. 이것은 장점 (프로세스 엔진에 대한 실험을 할 때)도 될 수 있지만 부담(대상 소프트웨어 프로세스에만 집중하려고 할 때)이 될 수 도 있다.

```
distribute_review_package (?p:rev_product) :
( and (forall DEVLOPER ?d
        suchthat (member [?p.reviewers ?d])
     exists DOC ?rp
        suchthat (linkto ?p.review_pkg ?rp

:

{DISTRIBUTE mail ?d ? rp}
  (?p.review_status = Ready);
  (?preview_status = notReady);
```

Example Rule

```
REV_PORODUCT :: superclass PRODUCT;
  ...
  reviewer:set_of DEVELOPER;
  review_pkg: DOC;
  review_status:(NotReady, Ready, Passed,
                 Failed);

MODULE:: superclass REV_PRODUCT;
...

DESIGN_UNIT :: superclass
REV_PRODUCT;
...

DEVELOPER:: superclass PERSONNEL;
....
```

Partial Product Model

(그림 3.4) MARVEL에서의 검토 프로세스

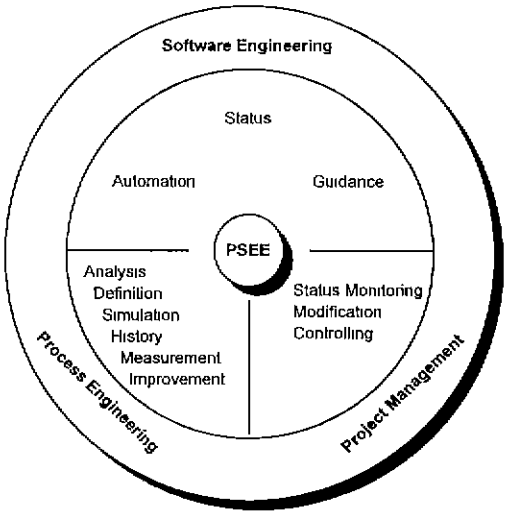
4. S/W 프로세스 자동화 지원 환경

4.1 프로세스 중심 통합환경의 요소

프로세스 모형화의 연구가 진전되면서, 프로세스 모형화 언어 자체보다는 소프트웨어 프로세스 실행 지원 기능을 소프트웨어 개발 환경에 통합시키는 것으로 연구의 초점이 옮겨가고 있다 [2].

이러한 결과로 대두된 것이 프로세스 중심 통합환경(PCE: Process Centered Environment)이다. 프로세스 중심 통합환경은 소프트웨어 개발 프로세스에 대한 정의와 정의된 프로세스의 활용을 지원하고, 프로세스를 소프트웨어 수명 주기상의 정적인 명세로만 다루지 않고 프로세스를 동적인 개체로 처리한다. 과거의 전통적인 환경과 프로세스 중심 통합환경과의 차이는 소프트웨어 개발 수명주기의 각 단계에 대한 지원의 범위에서 나타난다. (그림 4.1)에서와 같이 프로세스 중심 통합환경은 프로세스 공학, 프로젝트 공학, 소프트웨어 공학의 활동을 통합한 것이다. 프로세스 공학 활동은 소프트웨어 프로세스 모형을 정의하고

평가하는 활동이다. 프로젝트 관리 활동은 프로세스 공학에서 정의된 프로세스 모형을 선정하고 조정하고, 결정된 프로세스의 실행을 감독하는 활동이다. 소프트웨어 공학 활동은 프로젝트 관리에서 선정된 특정 프로세스를 실행하는 활동이다. 프로세스 중심 통합환경은 이러한 세가지의 활동을 지원한다.



(그림 4.1) 프로세스 중심 통합환경

4.2 프로세스 중심 통합환경의 특징

프로세스 중심 통합환경에서는 프로세스 모형을 이용하여 소프트웨어 프로세스의 설계, 분석, 자동화, 감시, 공유 등을 가능하게 한다. 현재까지 개발된 몇 가지의 프로세스 중심 통합환경의 공통적인 특징을 기술하면 아래와 같다.

(1) 프로세스 정의

프로세스 공학자는 프로세스 중심 통합환경을 이용하여 프로젝트의 프로세스를 정의한다. 3장에서 설명한 여러 가지의 방법을 이용하여 프로세스를 정의한다. 예를 들면 ETVX, IDEF, Statemate, Petri net 등이 있다.

(2) 프로세스 분석

프로세스 중심 통합환경의 프로세스 모형은 일치성, 완전성, 정확성 등에 대한 분석이 가능하다. 예를 들면 프로세스의 동시에 수행되는 활동의 최대값 (일정관리 목적) 이나 산출물이 사용되고 있지 않은 활동의 식별 등의 분석이 수행된다.

(3) 프로세스 설명

프로세스 중심 통합환경은 프로세스 및 제품 정보에 대한 그래픽 디스플레이를 지원하여 프로젝트 관련 조직 구성원간의 의사 소통을 원활하게 한다.

(4) 프로세스 모의실험

대규모의 소프트웨어 프로세스를 실행하는 것은 많은 자원이 소요된다. 프로세스 중심 통합환경에서는 프로세스를 실제로 실행하기 이전에 평가할 수 있도록 하는 모의 실험 기능이 제공되어 있다.

(5) 프로세스 자동화

소프트웨어 프로세스 중에서 사람의 개입이 필요하지 않은 활동들은 자동화가 가능하다. 이러한 활동의 예로는 최근에 변경된 모듈에 대한 회귀시험의 수행이다. 변경에 관련이 있는 사람들에게 통보하는 활동도 자동화가 될 수 있는 활동이다. 프로세스 중심 통합환경에서는 프로세스가 정의되면 이러한 통보 활동이 식별되고 자동화된다.

(6) 프로세스 감시

프로세스 중심 통합환경의 가장 중요한 특징은 프로세스의 실행을 감시하고 수행된 활동의 이력을 기록하는 것이다. 프로세스의 이력은 미래의 프로세스 개발과 개선에 사용될 수 있다.

(7) 프로세스 변경 지원

소프트웨어 프로세스의 변경은 새로운 기술의 발전이나, 새로운 프로세스 패러다임의 개발 등의 이유로 발생한다. 프로세스 중심 통합환경을 채택한 조직에서는 조직의 작업을 중단시키지 않고

프로세스 정의를 변경할 수 있어야 한다.

(8) 개방성(Openness)

프로세스 중심 통합환경은 기존의 도구들 사이의 제품 데이터 교환을 위한 import/export 기능을 제공한다.

(9) 다중 사용자 지원

프로세스 중심 통합환경은 일반적으로 많은 사용자를 가정하고 있다. 한 명의 사용자라고 하여도 그 사람은 프로세스 공학자, 관리자, 소프트웨어 공학자 등의 여러 가지 역할을 수행하여야 한다. 따라서 프로세스 중심 통합환경은 하나의 프로세스에 여러 명으로 구성된 팀의 활동을 지원할 수 있도록 되어 있으며, 프로세스와 제품 자료에 대한 접근 제어 기능도 있다.

(10) 프로세스 안내(Process Guidance)

프로세스 중심 통합환경은 소프트웨어 공학자에게 모형화된 프로세스와 현재의 상태를 근거로 하여 가능한 다음 단계를 알려준다.

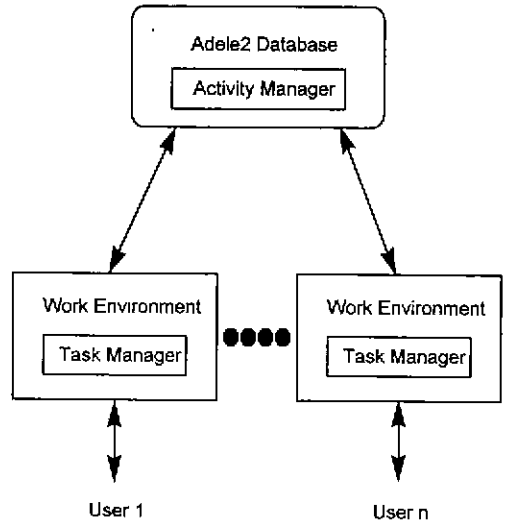
(11) 작업 중심 사용자 인터페이스

모형화된 프로세스에 근거하여 프로세스 중심 통합환경은 사용자에게 현재 작업에 적합한 정보와 도구만을 제공할 수 있다. 이것은 사용자에게 불필요한 정보를 주지 않고 현재의 작업에서 유용한 것만을 제공하여 생산성 향상과 이용 편리성을 제고할 수 있게 한다.

4.3 프로세스 중심 통합환경 도구

(1) Adele2

Adele2는 프랑스에서 개발된 프로세스 중심 통합환경이다. (그림 4.2)에 Adele2의 구조가 제시되어 있다. 프로젝트 팀은 하나의 Adele2 데이터베이스를 공유한다. 이 데이터베이스는 논리적으로 집중되어 있고 물리적으로는 분산되어 있다. 활동관리자는 복수개의 작업환경의 활동을 조정한다. Adele2의 액티브 데이터베이스는 객체지향 데이터 모형을 이용하여 객체를 정의하고 사건과 촉발(event and trigger)을 이용하여 활동을 기술한다.



(그림 4.2) Adele2의 구조

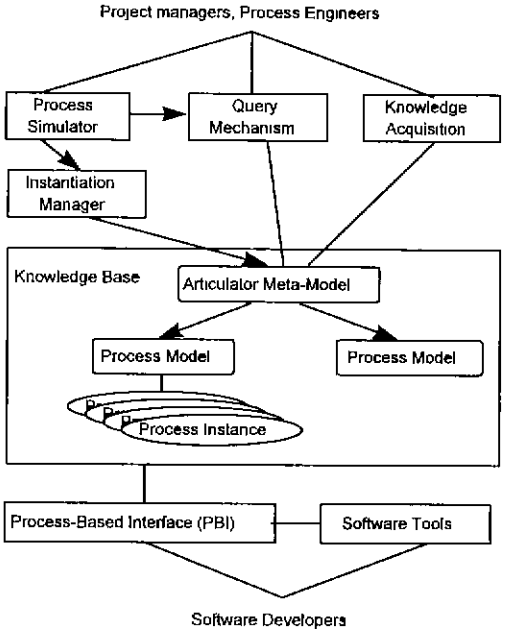
(2) Arcadia

Arcadia는 미국 국방부에 의하여 지원된 연구 컨소시엄이다. Arcadia의 목표는 유연하고 확장가능하고 통합된 소프트웨어 공학 환경을 개발하는 것이다. 이러한 목표를 달성하기 위하여 프로세스 프로그램을 사용한다. Arcadia의 특징은 객체 관리와 사용자 인터페이스 관리외에 프로세스 프로그램 인터프리터를 갖고 있다는 것이다. Arcadia의 프로세스 프로그래밍 언어는 Ada 프로그래밍 언어를 관계형으로 확장한 것으로 APPL/A라고 불려진다. APPL/A를 이용하여 프로세스 프로그래머는 여러 가지 제품간의 관계와 관련 제품들간의 제약조건을 표현할 수 있다.

(3) Articulator

Articulator는 소프트웨어 프로세스 모형화 및 분석에 지식베이스 방법을 적용한 것이다. 여기에서 프로세스 모형은 객체 인스턴스 네트워크로

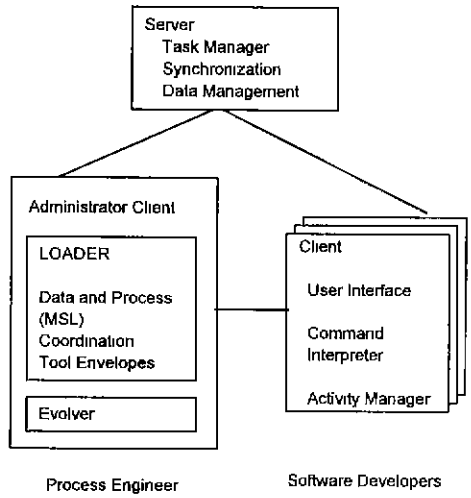
표현된다. Articulator는 메타모형화 개념을 지원한다. 메타모형화란 프로세스 공학자가 시스템의 프로세스 모형화 개념을 변경할 수 있는 개념이다. (그림 4-3)에 Articulator 시스템의 구조가 제시되어 있다. 프로젝트 관리자와 프로세스 공학자는 프로세스 모의실험기, 질의 메카니즘, 지식 획득 모듈 등을 통하여 Articulator와 상호 작용한다. 지식 베이스는 메타모형, 몇 가지의 프로세스 모형, 그리고 이것들의 인스턴스를 포함한다. 프로세스 모형의 인스턴스화는 프로세스 기반 인터페이스 이용하여 이루어진다. Articulator는 제안된 프로세스 모형에 대한 모의실험을 할 수 있으며 문제가 발생하면 이를 해결할 수 있는 수단을 제시할 수 있다.



(그림 4.3) Articulator 시스템의 구조

(4) MARVEL

MARVEL은 최초의 프로세스 중심 통합환경 중의 하나이다. 이것은 제품 정보에 대한 객체지향 표현과 프로세스 정보에 대한 법칙 기반 표현을 제공한다. 프로세스 법칙도 객체지향 방법으로 구성된다. 프로세스 법칙은 MARVEL Strategy Language (MSL)라는 특수 언어로 기술된다. (그림 4.4)에 Marvel 시스템의 구조가 나타나 있다.



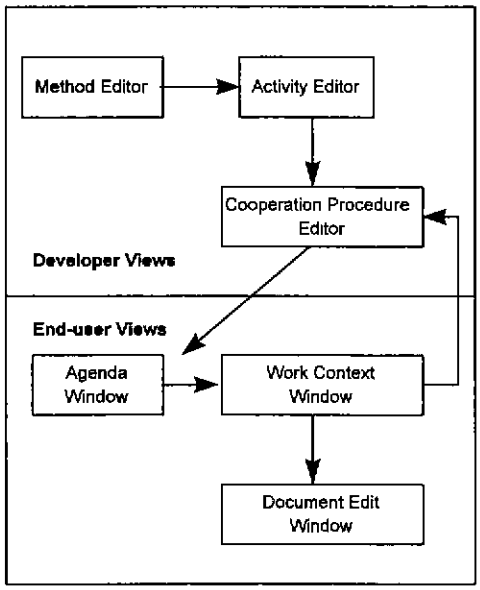
(그림 4.4) MARVEL 시스템의 구조

서버 프로세스는 프로젝트 팀에 대한 정보를 유지하고 작업관리와 접근 동기화 기능을 제공한다. 소프트웨어 개발자인 클라이언트는 서버에 연결된다. 프로세스 공학자는 데이터 모형과 프로세스를 관리 클라이언트를 이용하여 정의한다.

(5) ProcessWEAVER

ProcessWEAVER는 상용 시스템으로, 개별 작업과 프로세스 자동화 (즉 일련의 작업)를 관리하는 도구를 제공한다. (그림 4.5)에 Process-WEAVER의 구조가 제시되어 있다. Process-WEAVER의 사용자는 agenda라는 주 화면을 통하여 작업 (Work Contexts라고 부름)을 수행하거나, 보내고, 받는다. Method editor는 활동의 계층을 정의할 수 있고 Activity editor는 활동에

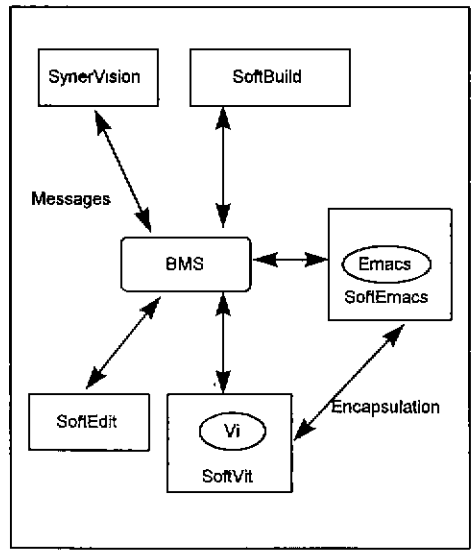
대한 입력, 출력, 역할을 정의할 수 있게 한다. Cooperative Procedure editor를 통하여 각각의 활동에 대한 세부 작업수준의 프로세스를 페트리 네트의 방식을 이용하여 모형화할 수 있다. 사용자가 프로세스를 수행할 수 있도록 하는 윈도우는 work context editor이다.



(그림 4.5) ProcessWEAVER의 구조

(6) SynerVision

SynerVision은 상용 시스템으로 HP의 SoftBench 소프트웨어공학 환경하에서의 프로세스 감시 및 실행 도구이다. SynerVision의 구조는 (그림 4.6)과 같다. SynerVision은 Broadcase Message Server (BMS)와 파일시스템을 통하여 SoftBench 환경에 통합되어 있다. SynerVision은 BMS를 이용하여 콘트롤을 통합하고 데이터 통합은 파일시스템을 통하여 이룬다. SynerVision에서 프로세스는 일련의 작업이다. 작업은 제목, 사전에 정의된 속성 및 사용자 정의 속성, 일련의 행위들, 종속관계를 갖는다. 작업의 속성은 작업에 대한 특성 정보를 유지하게 한다. 작업의 특성 정보에는 작업의 소유자, 작업의 상태, 완료 예상 시간, 실제 작업 시간 등이다. 프로세스 공학자는 SynerVision에서 이 밖의 작업 속성을 정의할 수 있어서 조직의 특성에 맞는 메트릭 데이터를 수집할 수 있다.



(그림 4.6) SynerVision의 구조

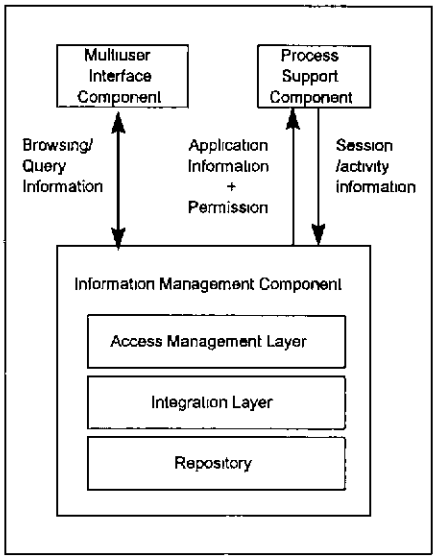
(7) 기타

이밖에도 Melmac, Matisse, Merlin, SPADE 등의 프로세스 중심 통합환경이 있다.

4.4 프로세스 중심 통합환경의 정보 관리

프로세스 중심 통합환경의 구조는 (그림 4.7)과 같이 다중 사용자 인터페이스, 프로세스 지원 요소, 정보 관리 구성 요소의 세개 요소로 구성된다.

다중 사용자 인터페이스는 여러명의 분산된 사용자를 지원하는 기능이고 프로세스 지원 요소는 하나 이상의 프로세스 모형화 포말리즘을 제공하는 프로세스 실행 엔진을 가지고 있다.



(그림 4.7) PCE의 구조

프로세스 중심 통합환경은 프로젝트의 소프트웨어 개발 프로세스를 모형화하여 실행할 수 있게 하는 지원 기능이 있는데, 실행 기능은 단순하게 개발자가 프로세스를 따라갈 수 있도록 인도하는 기능에서부터 실제로 모형을 실행하는 기능까지 다양하다. 이러한 기능을 수행하기 위하여 프로세스 중심 통합환경은 개발중인 프로젝트에 대한 방대한 데이터를 일관성 있고 질의를 지원할 수 있도록 관리해야 한다. 이러한 관점에서 프로세스 중심 통합환경은 데이터베이스 응용 시스템과도 유사하나, 프로세스 중심 통합환경은 데이터 관리 요구가 데이터베이스 응용 시스템과 다음과 같은 면에서 다르다. 첫째는 프로세스 중심 통합환경은 형태가 매우 상이한 데이터를 관리한다는 점이다. 두 번째의 다른 점은 프로세스 중심 통합환경은 장기간 지속되고, 상호작용적인 활동들이 데이터를 액세스한다는 것이다.

이러한 데이터 관리 요구를 만족시키는 데이터베이스를 구축하기 위하여 관계형 데이터베이스와 객체지향 데이터베이스를 응용하여 왔으나, 어떠한 상용 시스템도 프로세스 중심 통합환경의 데이터 관리 요구를 만족시키지 못한다.

프로세스 중심 통합환경을 위한 정보 관리는 다음의 두 가지 형태로 발전될 것이다. 첫 번째, 특정 객체지향 데이터베이스를 확장하여 프로세스 중심 통합환경의 정보 요구를 만족할 수 있게 하는 것이다. 두 번째의 방향은 개방형 저장소(open repository) 방법이다. 이것은 서로 다른 프로세스 중심 통합환경의 데이터 관리 요구를 모두 만족하는 데이터베이스 구조를 만들지 않고 정보 관리 요소가 상호 운용될 수 있는 체계를 만드는 것이다.

5. 결론 및 향후 발전방향

본고에서는 프로세스 자동화의 개념 및 필요성을 설명하고, 자동화를 위한 요소기술들인 프로세스 모형화 기술 및 프로세스 중심 통합환경에 대하여 고찰하였다.

1980년대 중반에 IBM이 소프트웨어 개발 프로세스 자동화 연구를 시작한지 10년만에 시장에서 프로세스 자동화에 대한 소개 제품을 보게 되었다. 소프트웨어 프로세스 자동화는 사업적 요구를 포함하여 하드웨어의 저가격화 경향, 통신이나 통합 표준에의 순응성 증가, 효과적인 프로세스 중심 통합환경 기술의 발전에 따라 더욱 발전할 것으로 보인다. 또한, 소프트웨어 프로세스 개선이나 비즈니스 프로세스 재공학에 요구가 강조되면서 소프트웨어 프로세스 자동화 방향으로의 이전을 부추킬 것으로 보인다.

미래의 응용개발에서 프로세스 중심 통합환경은 지금 인식하는 것보다 더 광범위한 지원을 제공할 것이다. 관리되는 프로세스의 현재 상태에 대해 프로세스 중심 통합환경이 이해하고 있어 이러한 지식이 응용영역의 지식과 연결된다면 뚜

렷한 상승효과가 발생할 것이다. 특히, 아래 분야를 위한 응용영역/작업에 지원이 가능할 것이다.

- 재사용 컴포넌트의 선택
- 적절한 도구 또는 기법의 선택
- 적절한 알고리즘의 선택
- 테스트 Suites의 선택
- 목표 환경에 정보 제공
- 업무를 통한 초보자 안내
- 계획상의 일정과 실제 일정의 비교, 제안
- 메트릭 분석의 자동화 수행
- 프로세스 개선 노력을 지원하는 안내 제공

이러한 분야에 규칙기반 추론(Rule-based Reasoning)을 사용하는 것이 자연스러우며, 특히 규칙이 이미 프로세스 모형을 드라이브 하기 위해 사용되는 경우에는 더욱 그렇다.

결론적으로 소프트웨어 프로세스 자동화가 잠재적으로는 소프트웨어 생산성 및 품질에 극적인 개선을 가져다 줄 수 있으나, 해결해야 할 사항들이 아직 남아있어 그에 대한 연구가 필요하다.

먼저, 프로세스 모형화를 보면, 인간 및 조직의 행위가 소프트웨어 개발작업에 미치는 영향 및 관계의 모형화에 대한 연구가 더욱 진행되어야 하며, 그 외에 보다 소프트웨어에 적합한 모형의 개발, 모형화 언어의 여러 패러다임의 통합, 모형의 분석 및 검증을 위한 추가적 방법, 모형화의 사용자 편이성 제고 방법 등이 연구되어야 할 것이다. 프로세스 중심 통합환경과 관련하여서는 표준화, 통합, 개방성, 호환성, 사용편이성 제고 등이 실제적인 면에서 필요하며 초기 프로세스의 개발, 다중 사용자 인터페이스 설계 등에 대한 연구가 필요하다.

참고문헌

[1] 양해술, "프로세스 모델 기반 개발방법과 프로세스의 평가", 정보과학회지, 제13권9호, 1995.9

[2] A. Fuggetta and A. Wolf edition, Trends in Software: Software Process, Wiley and Sons, 1996.

[3] Bandinelli, S., A. Fuiggetta, and S. Grigolli, "Process Modeling in the Large with SLANG," In Proc. 2nd Int'l Conf. on the Software Process, IEEE CS Press, 1993, 75-83.

[4] Belkhatir, J. Estublier, and W. Melo, "ADELE-TEMPO: An Environment to Support Process Modeling and Enaction," Software Process Modeling and Technology, 1994, 187-222.

[5] Christie A.M., Software Process Automation, Springer, 1995.

[6] Fernstrom, C. "Process Weaver: Adding Process Support to UNIX," In Proc. 2nd Int'l. Conf. on the Software Process, IEEE CS Press, 1993, 12-26.

[7] K.D. Saracelli and K.F.Bandat, Process Automation in Software Application Development, IBM systems Journal, Vol.32, No.3, 1993

[8] McGowan, C., and S. Bohner, "Model Based Process Assessments," In Proc. 15th Int'l Conf. on Software Eng., IEEE CS Press, 1993, 202-211.

[9] Paul, M., B. CUrtis, M.B. Chrissis, and C. Weber, "Capability Maturiy Model, Version 1.1," IEEE Software 10:4 (July 1993), 18-24.

[10] Toshfumi Tanaka, et al., "Improvement of Software Process by Process Description and Benefit Estimation", In 17th Int'l. Conf. on Software Engineering, April, 1995

안 유 환

1984년 서울대학교 산업공학과 졸업 (학사)
1986년 한국과학기술원 경영과학과 졸업 (석사)
1990년-현재 한국과학기술원 경영과학과 박사과정
1986년-현재 시스템공학연구소 선임연구원
1996년-현재 시스템공학연구소 품질관리 연구실장
관심분야 : S/W 품질보증, S/W 품질평가, S/W 프로세스 관리 및 자동화, S/W 프로세스 개선, 개발/관리 방법론, Client/Server 시스템 개발방법론

이 양 규

1985년 고려대학교 경영대학 경영학과 졸업 (학사)
1987년 한국과학기술원 경영과학과 졸업 (석사)
1992년 한국과학기술원 경영과학과 졸업 (박사)
1992년-1995년 국방정보체계연구소 선임연구원
1995년-현재 서원대학교 경영정보학과 조교수
관심분야 : 페트리네트 응용, 실시간 시스템 모델링, 소프트웨어 개발 방법론, 소프트웨어 개발비 산정, 소프트웨어 품질 평가 및 개선

이 승 지

1992년 아주대학교 전산학과 졸업 (석사)
1983년-현재 시스템공학연구소 선임연구원
1991년-92년 미국 IBM Stelingforest 객원연구원
관심분야 : 프로젝트 관리, S/W 개발생산성 측정 및 비용예측, S/W 프로세스 관리, S/W 품질관리

김 길 조

1987년 서울대학교 산업공학과 졸업 (학사)
1989년 한국과학기술원 경영과학과 졸업 (석사)
1997년-현재 시스템공학연구소 연구원
관심분야 : S/W 품질평가, S/W 프로세스 관리 및 자동화, S/W 개발방법론, 시뮬레이션