신기술해설

소프트웨어 프로세스 리엔지니어링(SPR)의 개요와 접근 방법

양 해 술† 이 용 근†† 황 인 수†††

❖ 목 차 ❖

1. 서 언

최근에 소프트웨어 개발 조직을 체계적으로 개선하는 SPR(Software Process Re-engineering)의 개념과 방법에 대해 많은 관심이 집중되고 있다. 즉, 프로세스 프로그래밍의 제안을 계기로 하여 소프트웨어 개발 순서를 형식적으로 기술한 소프트웨어 프로세스에 관한 연구와 개발이 시작되었다. 프로세스는 프로덕트와 함께 소프트웨어 개발이라고 하는 자동차의 양바퀴를 구성하는 본질적인 개념으로, 프로세스의 연구와 개발은 생산성과 품질의 향상, 개발기간의 단축을 꾀한다는 점에서 새로운 기술로 다가오고 있다. 그러나 프로세스의 실행 주체가 사람이기 때문에 프로세스의 개선에는 기술적인 문제이외에도 모든 인간적 요인 등의 많은 문제가 잠재되어 있으므로 이를 고려하여야 한다.

반면에 기업경영에서는 BPR(Business Process Re-engineering)이 주목받고 있으므로 소프트웨어 개발 조직에서도 SPR의 개념을 도입하여 체계화할 필요가 있다. BPR과 SPR은 둘다 프로세스에 착안하여 작업 계획을 개선하는 방법이기 때문에 서로 공통점이 많다고 볼 수 있다. 따라서 본 고에서는 다른 분야에서의 프로세스 개선방법과도 연관시켜 SPR의 개요와 적용 방법에 대해 기술하기로 한다.

2. SPR의 개요

2.1 SPR의 개념

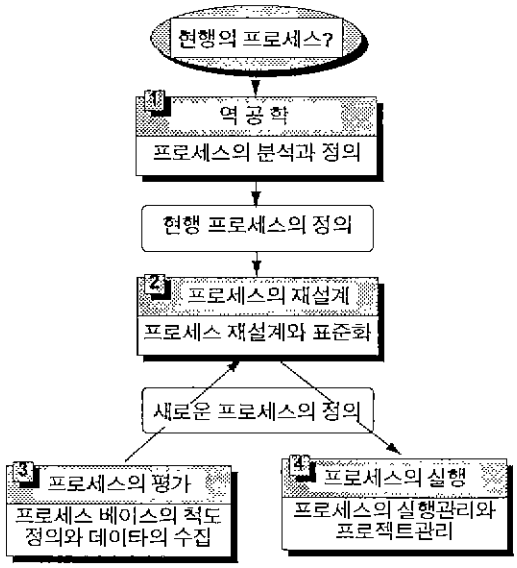
SPR이란 프로세스에 준하여 개발 계획을 수정하고 생산성과 품질을 향상시키는 방법이라고 볼 수

† 종신회원:한국소프트웨어품질연구소(INSQ) 소장
†† 종신회원:강원대학교 전자계산학과 박사과정
††† 정 회 원.삼성데이타시스템(주) 정보기술연구소 수석

있다. 따라서 SPR은 형식적 프로세스 정의에 기초한 소프트웨어 시스템의 설계와 같은 설계방법론으로 경영을 지배하고 체계적인 프로세스의 재구축을 도모한다는 점에서 전사적 품질관리(TQC) 등 종래의 프로세스 개선방법을 진일보 발전시킨 것이라고 할 수 있다.

2.2 SPR의 프로세스와 방법

SPR에서는 우선 현행의 프로세스를 명확히 정의한 다음에 재구축하는 방식을 채택하는 것으로서 SPR 프로세스 접근 방법의 과정을 (그림 1)에 나타내었다.



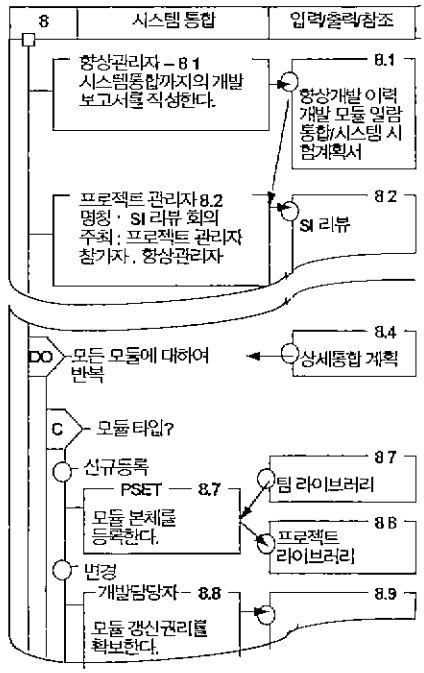
(그림 1) SPR의 프로세스 접근 방법

(1) 프로세스의 역공학

우선 현재의 작업을 있는 그대로 눈에 보이는 형태로 나타내는 것으로부터 출발점이 된다. 즉, 역공학이란 현재의 작업내용에서 그 프로세스를 명확히 하여 문서화하는 것이라고 볼 수 있다.

일반적으로 작업내용은 조직 전체 또는 프로젝트마다 작업표준으로 문서화되어 있다. 그러나 작업표준은 작업의 상세한 구조가 명확히 규정되어 있지 않다든가 형식적이지 못하다는 문제점이 있기 때문에 프로세스는 소프트웨어의 설계와 마찬가지로 형식성과 추상화의 구조를 가진 프로세스 기술언어로 기술된다. (그림 2)와 같이 트리구조 차트를 확장시킨 프로세스 기술언어를 사용하여 프로세스를 정의하고 있으며 이 밖에 데이타 흐름도나 객체지향 분석·설계 방법론의 기술방법 등의 많은 프로세스 기술언어가 제안되어 있다.

이와 같이 사람에 의한 프로세스 정의를 컴퓨터에 의해 실행되는 프로세스 프로그램과 구별하여 프로세스 스크립트라고 부르고 있다.



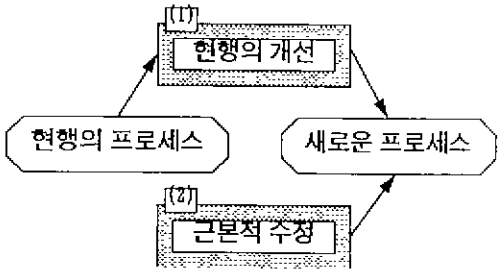
(그림 2) 프로세스 스크립트의 기술 예

(2) 프로세스의 재설계

이 부분에서는 현재의 프로세스를 기업 전략과 개발 프로젝트의 목표에 준하여 재설계하는 것이다. 여기에서는 (그림 3)에 나타낸 바와 같이 2가지의 접근 방법이 있다.

① 개선:현행 프로세스의 문제점을 소집단 활동 등을 통하여 분석, 개선하는 방법이다. 이 방법은 지속적 프로세스 개선(Continuous Process Improvement)으로 알려지고 있다.

② 근본적 수정:프로세스가 당연히 이상형에서 출발하는 프로세스의 개혁(Process Innovation)이 있다.



(그림 3) 프로세스 재설계의 2가지 접근 방법

위의 어느 접근방법을 채택할 것인가는 개발 프로젝트의 성질과 목표에 따라 달라지게 된다.

예를 들면 신규 프로젝트에서는 현행의 작업표준과 프로세스를 근본적으로 수정하여 개발전략에 적합한 새로운 프로세스를 설계할 수 있다. 반면에 기능추가와 같은 보수위주의 프로젝트에서는 프로세스 개선의 접근방법이 위험 절감 등의 점에서 바람직하다.

프로세스를 개선하기 위한 착안점의 하나는 재작업(Rework)의 예로서 AT&T나 HP의 조사에서는 재작업의 비율이 20~33%를 점유하고 있다. Raytheon에서는 검토 등의 방법을 개선하고 재작업 비율을 40%에서 11%로 절감시킨 결과 전체 30%의 비용절감을 달성한 사례가 보고되고 있다.

(3) 프로세스의 실행

프로세스의 실행 주체가 사람이기 때문에 프로세스의 실행과 관리가 매우 어렵다고 볼 수 있다. 또 다른 한편으로는 개발 비용과 납기를 엄밀하고 정확하게 관리해야 된다는 요구가 높아지고 있다. 종래의 프로젝트 관리방법은 진척과 공수 등을 비교적 거시적으로 관리하고 있기 때문에 이와 같은 요구에는 충분히 대응하지 못하고 있다. 그러나 프로세스에 기초한 접근방법은 다음과 같은 새로운 해결 방법을 가져왔다.

① 정밀도:개발의 최소단위인 개인마다 실행, 관리를 지원할 수 있다.

② 일관성:개인에서 팀, 프로젝트 전체에 걸친 일관된 공통의 척도가 된다.

③ 공통성:개인의 배후에 있는 공통의 구조를 나타내는 것으로 참조할 수 있는 기준이 된다.

더우기 개인마다 프로세스의 실행을 지원하기 위해 프로세스에 준한 도구를 통합하는 CASE환경이 개발되어 있다. 그리고 그룹웨어중에서도 작업흐름 관리로서 프로세스의 실행을 지원하는 환경이 제공되어 있다. 또한 프로젝트를 관리 지원하기 위해 프로세스의 실행에 따라 각종 개발 데이타를 자동적으로 수집하고 계획과 관리를 지원하는 환경도 개발되고 있다.

(4) 프로세스의 평가

프로세스의 평가 방법을 분류하면 〈표 1〉과 같다.

〈표 1〉 프로세스 평가의 접근

구 분	절대 평가	상대 평가
매크로 평가	CMM	벤치마킹 평가
마이크로 평가		

① 매크로 평가와 마이크로 평가

(a) 매크로 평가의 방법

기업과 프로젝트 전체의 프로세스의 양호성을 평가하는 방법. 예를 들면 프로세스 성숙 모델 CMM(Capability Maturity Model)은 5단계의 척도로 평가한다. ROI(Return On Investment:투자 대 효과)나 프로젝트의 총 비용은 기업 경영의 관점에서 평가하는 정량적인 척도이다.

(b) 마이크로 평가 방법

프로세스 실행 데이타의 분석과 개발자의 의견 등으로부터 공정과 팀마다의 문제점을 찾아내는 방법이다. 예를 들면, 개인마다 작업 시간 분석에 의해 모든 작업 시간의 60%는 물품과 작업의 조정 등 비생산적인 작업에 소비하고 있는 것이 명확해졌다.

② 절대평가와 상대평가

(a) 절대평가 방법

CMM과 총비용 등 절대척도에 준하여 평가하는 방법이다.

(b) 상대평가 방법

다른 기업에서 수행하고 있는 프로젝트와 과거의 데이타를 비교하여, 평가하는 방법이다. 벤치마킹(Benchmarking)은 타사나 타프로젝트의 프로세스와 비교하여 베스트프렉티스(Bestpractice)라고 불리는 실천으로 성과를 올리고 있는 우수한 기술을 발굴하는 방법으로서 소프트웨어 개발에도 적용하기 시작하였다. 절대평가와 상대평가는 보완적 관계에 있으므로 조합시켜 이용하면 더 효과가 있다.

2.3 소프트웨어 프로세스 평가 모델

1980년대 후반부터 소프트웨어 프로세스가 가지는 총합적인 능력을 직접적으로 평가, 개선하도록 하는 경향이 확대되고 있다. 대표적인 소프트웨어 프로세스 평가 모델로 CMM과 SPICE, 기타 모델에 대하여 설명하기로 한다.

(1) CMM(Capability Maturity Model)

CMM은 미국 카네기대학 소프트웨어기술연구소(SEI)에서 1987년 제창된 프로세스 성숙도 모델로 현재도 SEI에서 개량하기 위해 연구개발을 행하는 동시에 보급 활동을 계속하고 있다. CMM은 조직, 작업 순서, 관리방법, 기법, 도구, 환경 등을 종합적으로 포함한 것을 프로세스로 정의하고 프로세스를 실행하는 능력이 우수할수록 보다 좋은 제품과 서비스가 가능하다고 하는 사고방식을 기초로 하여 능력 성숙도 모델을 구축하고 있다. CMM에서는 프로세스의 성숙도를 Initial, Repeatable, Defined, Managed, Optimized의 5단계로 분류한다.

① 레벨 1:Initial

공식적인 절차는 거의 없으며, 개발자는 편한대로 여러 작업을 할 수 있다. 비록 규격이 있더라도 무시되고 도구가 있어도 사용되는 것은 별로 없다. 생산효율은 낮으며 견적을 추출하는 것도 어렵다. 외부에서는 개발자가 현재 무엇을 실행하고 있는지를 좀처럼 이해할 수 없다.

② 레벨 2:Repeatable

조직에 있어서 과거의 경험을 기초로 의견이 같을 수 있다. 같은 의견에 따라 과거와 같은 방법으로 작업을 처리하는 것이 가능하다. 레벨 1에 비해서 생산성은 다소 상승하고, 견적도 통계적으로 허용된 범위 내에서 추출이 가능하다. 그러나 각 작업은 아직 블랙박스로 되어 있고 프로젝트 관리자의 능력에 많은 부분을 의존하고 있으며, 만약 관리자가 프로젝트 도중에 빠진다면 프로젝트는 붕괴된다.

③ 레벨 3:Defined

작업과 작업 결과가 문서로 정의되고 문서에 의해 관리가 가능하다. 작업 결과는 외부로부터 문서를 통하여 파악할 수 있다. 프로세스를 문서화하는 것으로 프로세스의 개선이 용이하게 처리 될 수 있다.

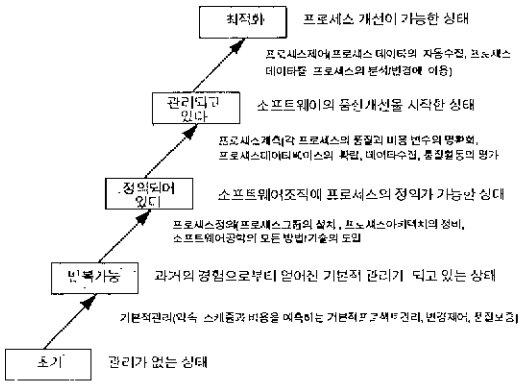
④ 레벨 4:Managed

각 프로세스에 대해서 매트릭스를 이용한 상세한 측정이 행해진다. 이것에 의해 외부로부터 조직의 현 상태를 정확하게 파악하는 것이 가능하고 품질의 향상에 커다란 도움이 된다.

⑤ 레벨 5:Optimized

이 단계에까지 도달한 조직에서는 항상 프로세스의 개선이 행해지고, 조직은 최적의 프로세스에 접근할 수 있다. 생산성은 대단히 높고, 견적 정도도 정확하다.

이와 같이 조직이 성숙함에 따라 레벨을 단계적으로 끌어 올릴 수 있으며, 단계를 건너 뛰는 것은 기술적으로 불가능하다. 또한 새로운 조직이 높은 레벨에서부터 출발하는 것도 있을 수 없다. 레벨을 올리는 것으로 조직의 생산성이 향상되고, 견적 정도도 보다 정확하게 될 수 있다.



(그림 4) 프로세스 성숙도의 레벨

따라서 현재의 조직을 단계적으로 상위의 상태로 이행하는 것을 권장하고 있으며 (그림 4)에 프로세스 성숙도의 레벨을 나타내었다.

과거의 조사 결과에 따르면 조직의 대부분은 레벨 1이고, 레벨 4이상의 조직은 거의 없는 것으로 나타나고 있다(1991년까지의 조사에 의하면 미국의 조직의 81%가 레벨 1이고, 12%가 레벨 2, 7%가 레벨 3, 레벨 4 이상은 0%였다). 그러나 특히 미국의 기업을 중심으로 조직의 개선 작업을 진행하고 있으며 장래에는 보다 높은 레벨의 결과가 나오리라고 예상된다.

(2) SPICE(Software Process Improvement and Capability dEtermination)

SPICE는 소프트웨어 프로세스의 개선과 능력의 결정을 목적으로 하는 프로세스 평가 모델이고, 동 모델을 기초로 한 국제 표준화의 활동이 ISO/IEC JTC SC7/WG 10에서 미국, 유럽, 캐나다, 일본, 한국 등의 참가로 진행되고 있다. SPICE의 프로세스 모델은 소프트웨어 프로세스의 실행에 필요한 기본 작업을 기초로 하여 계층적으로 구성된다. 이 프로세스 모델에서는 프로세스 전체를 다음 5가지의 프로세스 카테고리 즉, 고객-공급자, 엔지니어링, 프로젝트, 지원, 조직 등으로 분류하고 있다.

SPICE는 CMM의 발전계보로서 연구가 진행되고 있으며 다음과 같은 6 단계의 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

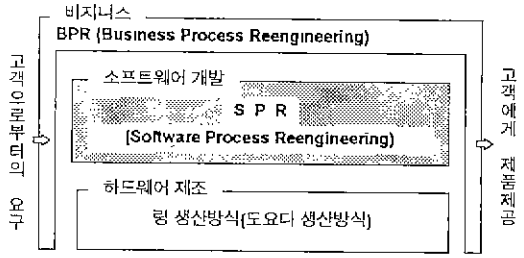
(3) 기타

Bootstrap은 유럽의 ESPRIT 프로젝트에서 연구되고 있는 소프트웨어 프로세스 평가방식이며, ISO 9000-3은 하드웨어 및 시스템의 개발에 관련된 표준인 ISO 9001을 소프트웨어의 개발 프로젝트에 적용하기 위한 가이드로 개정이 추진되고 있다.

2.4 프로세스 리엔지니어링의 계보

프로세스에 기초한 개선의 접근 방법을 프로세스 리엔지니어링이라고 부르며 프로세스 리엔지니어링은 대상 영역에 따라 (그림 5)와 같이 정리할 수 있다. 하드웨어의 제조에서는 생산 시스템 공학 중에서 프로세스 설계(Process Design) 방법이 개발되고 있다. 예를 들면, 림 생산 방식이라고 불리는 도요다 생산 방식은 프로세스의 관리와 개선에 주안점을 두고 있다. 도요다 생산 방식에서는 사람이 작업하는 장소에서의 문제를 프로세스의 구조 문제로 받아들이고 있으며 이와 같은 점은 소프트웨어 개발에서도 적용할 수 있다고 생각된다.

또한 개발 착수 후의 SPR의 효과에는 스스로의 한계가 있으므로 개발 이전의 비지니스 프로세스와 출하 후의 보수 프로세스를 포함하는 종합적인 프로세스 리엔지니어링으로 시점을 넓힐 필요가 있다.



(그림 5) 프로세스 리엔지니어링의 계보

3. SPR의 실천

3.1 프로세스 개선

(1) 프로세스 개선의 배경과 목표

프로세스 개선의 발단은 분산 병행 개발이라고 불리는 새로운 소프트웨어 프로세스의 도입에 있다고 볼 수 있다. 지역적으로 분산된 복수의 개발 거점에서 단일의 소프트웨어 시스템의 복수 기능을 병행하여 개발한다.

분산 병행 개발에는 프로세스의 관점에서 다음과 같은 문제가 존재한다.

① 분산 개발의 문제:개발 거점간에서 프로세스를 분석한다. 또한 개발 거점마다 프로세스나 진척 기준 등이 서로 어긋나 있다.

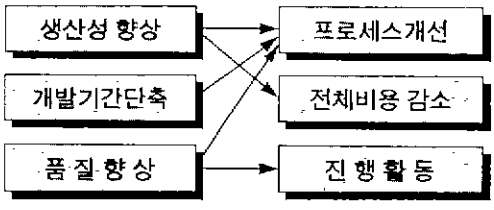
② 병행 개발의 문제:복수의 병행 개발 팀간의 프로세스 상호작용이 개발의 원활한 시행을 방해한다. 어떤 팀의 늦은 진척이 다른 팀의 진척을 곤란하게 한다던가, 시스템 통합 단계에서는 프로젝트 전체의 작업이 같은 시기에 마무리되는 것을 어렵게 한다. 분산 병행 개발을 전개할 때에 이와 같은 문제는 현장의 지혜와 프로세스 관리 등으로 대응하고 있다. 그러나 프로세스 프로그램의 제안과 그 후의 프로세스에 관한 연구에 자극을 받아 프로젝트 공통의 프로세스 스크립트를 정의하고 그것에 준하여 개발을 진행하게 된다.

이와 같은 내용은 한국소프트웨어품질연구소(INSQ)에서 프로세스에 기초한 공통의 척도에 의해 진척과 품질을 정량적으로 관리하는 방법과 지원 환경의 개발과 적용을 위한 체계적인 연구로 발전하고 있다. 현재에는 전 공정에 걸친 프로세스를 중심으로 하는 개발 품질보증과 품질평가 지원활동을 제공하고 있다.

특별히 서술해야 할 것은 집중 개발에서 분산 병행 개발로의 이행이 프로세스에 그치지 않고 개발 관리나 지원 환경 등 개발의 모든 면에서 개선을 촉구하고 있다는 점이다. 분산 병행 개발에서는 집중 개발에서 알 수 없었던 프로세스의 실행과 관리상의 문제가 드러나고 있다. 또한 문서 등의 프로덕트의 전자화도 촉진되고 있으며 도요다생산방식의 개발과 도입과정에서는 종래의 프로세스에서 은폐되어 있던 문제점이 드러나고 프로세스 개선의 계기가 되었다.

(2) 프로세스 기술 언어 YPL과 프로세스 정의

(그림 2)에 나타낸 바와 같이 전 공정에 걸친 프로세스를 트리구조차트의 하나인 YACII를 확장시킨 프로세스 기술언어 YPL(YACII-oriented Process Description Language)로 기술하였다. YACII를 채용한 것은 상세설계언어로서 일반적인 프로젝트에서 전면적으로 사용하고 있기 때문이다.



(그림 6) SPR의 접근

(3) 프로세스의 실행과 관리의 지원:PRIME

프로세스에 준한 개발지원환경 PRIME(PRocess Information ManagEr:프로그램)을 개발, 적용하고 있다. 개개인의 개발자에서 팀, 프로젝트에 이르는 각 계층 구조에 대응하여 프로세스의 실행과 관리를 일관성 있게 지원한다. PRIME은 1인 1대의 워크스테이션과 WAN(Wide Area Nwtwork)을 소개한 클라이언트/서버(Client/Server)형의 광역 분산 처리 환경을 전제로 하여 다음과 같은 기능을 제공한다.

① 프로세스 실행 지원:개개인의 개발자에 대하여 프로세스 스크립트의 제시

② 프로세스 관리 지원:개인, 팀, 프로젝트 각 계층에서의 프로세스 실행 정보의 수집과 평가, 공유에 의한 관리 지원

③ 도구와 설계 정보 등의 개발 자원의 통합

한편, PRIME에서는 프로세스의 주체가 사람이라는 점을 고려하여 다음과 같은 연구를 하고 있다.

(a)관리 수준의 설정과 진행의 허용

어느 수준보다 상세한 프로세스의 실행 순서는 임의로 한다. 이것은 기구 설계에서 진행의 개념과 대응하고 있다.

(b) 비동기성의 허용

프로세스의 실행은 똑같이 진행하거나 반복하기도 한다. 이와 같은 복수의 공정에 걸친 비동기성의 실행을 지원한다.

프로젝트 관리에서 종종 문제가 되는 것은 이와 같은 관리 수준이 명확하지 않기 때문에 무엇을 관리해야 하고 무엇을 관리하지 않아도 되는가를 알 수 없다는 점이다.

3.2 전체비용 절감

전체비용 절감이란 프로젝트마다 전체 개발비용을

감소시키는 방법의 체계이다. 프로세스에 기초하여 산정된 공정마다 표준 비용을 기준으로 하여 종래보다 자세하게 개발비용을 관리하고 동시에 감소시키는 것이다.

3.3 진행 활동

진행 활동이란 프로덕트의 품질에 관한 정량적 데이타에 준하여 품질 향상의 관점에서 프로세스를 개선하는 방법으로서 INSQ에서 전체적인 품질 평가 메트릭스를 개발하고 있다. 이 진행활동에 따라 프로젝트를 수행하면 프로세스를 개선할 수 있을뿐만 아니라 고품질의 소프트웨어를 얻을 수 있다.

4. SPR의 사례

소프트웨어 프로세스 리엔지니어링(SPR)의 몇 가지의 사례를 개선 실시조직, 주요한 개선성과, 개선방법에 대하여 〈표 2〉에 나타내었다. SPR을 적용하여 실시한 결과 비용감소와 품질향상 등의 개선이 있었음을 보여준다.

〈표 2〉 SPR의 사례

개선 실시 조직	주요한 개선 성과	시기 및 기관	개 선 방 법
Hughes 조립형시스템개발 (500인)	비용 감소 $200万/年 (예산초과율:6%→3%) ROI〉5 CMM Level 2→3	88년~90년	· 정량적 프로세스관리 · 프로세스 개선 팀 · 명세서결정에의 참가, 교육훈련 · 품질보증의 강화 검토개선
Raytheon 조립형 시스템개발 (400인)	비용감소 $1,580万/4年 (30% 감소) ROI=7.7 생산성 향상=2.3배 CMM Level 1→3	88년~92년	· 프로세스개선프로세스의 확립 · 프로세스의 표준화 · 재작업의 절감(Crosbydml 방법) · 검토의 형식화 · 명세의 조기확립
Motorola 휴대전화시스템 (1000인이상)	CMM Level 1→2	92년~93년	· 조직적인 개선의 계획 · 부문내에서의 프로세스 평가프 로세스 개선 프로세스의 정의와 진척평가
Hewlett-Packard (소프트웨어부문) (3,500인)	비용감소 $2,150万/年 (93년)	76년~93년	· 인스펙션에 의한 재작업의 절감
동서증권 (소프트웨어부문) 동서정보시스템	· 고품질 소프트웨어 개발 · 클라이언트 서버환경 · 라이트 사이징	94년~95년 평가기관:INSQ	· 정량적 품질평가 · 전사적 품질체계구축
대한무역진흥공사 (소프트웨어부문) KOTRA종합정보시스템	· 신뢰성 증가 및 비용감소	94년~95년 평가기관:INSQ	· 단계적 품질보증 · 정량적 품질평가
한진해운 (소프트웨어부문) 신재무정보시스템	· 고품질소프트웨어개발	95년~96년 평가기관:INSQ	· 전사적 품질보증 및 평가 관리 · 품질관리 체계 및 교육 훈련

5. SPR의 평가

SPR의 평가를 시작하는 것은 현시점에서는 곤란하지만 다음과 같은 점에서 소프트웨어 개발의 기초기술로서 받아들여지고 있다.

(1) 분산병행 개발의 실현

분산병행개발에 의해 개발사이클이 1년에서 3개월로 단축되고 있는 실정이다. 분산병행개발을 실행하는 열쇠는 프로세스의 정확한 파악과 실행의 관리에 있다고 본다. 실현 과정에서 프로세스의 구조가 중요하다는 것을 이해하고 실현하기 위한 기반기술이 되고 있다.

(2) 프로세스 중심 사고의 정착

프로젝트 내에서 문제가 발생했을때 항상 프로세스로의 피드백이 자연스럽게 논의되도록 되었다. 이것은 개인의 노력을 초월한 구조적인 개선을 촉구하는 것이다.

(3) 프로세스에 기초한 관리

종래의 프로젝트 관리는 일관성이 부족했다는 느낌을 부정할 수 없다. 프로세스에 기초하고 개인에서 팀, 프로젝트에 이르는 각 레벨에서 일관성 있는 관리를 실현하자는 것이다.

즉, 〈표 2〉에 나타낸 사례에서는 생산성, 총비용, 품질면에서 현저한 향상이 보고되고 있다. 특히 미국에서는 프로세스 개선의 평가척도로 총비용과 ROI 등 기업경영의 관점에서의 정량적 척도가 도입되었다. 이것은 생산성 향상 등의 성과를 기업경영과 관련하여 이해를 촉구한다는 점에서 유효하다. 반면에 이와 같은 평가 방법과 평가의 계획이 충분히 확립되어 있지 않다는 점에 유의할 필요가 있다.

따라서 SPR을 실현하기 위해 이론과 실제의 양면에서 연구, 개발, 적용, 평가를 시행할 필요가 있다. 즉, 이론적인 면에서는 SPR의 방법론을 개발할 필요가 있으므로 (그림 1)의 SPR의 각 프로세스에 걸친 방법의 연구, 개발 적용평가가 필요하다고 본다. 실용적인 면에서는 CMM의 보급과 더불어 미국에서 활동이 활발하다고 볼 수 있으며 BPR과 마찬가지로 성공사례 정도는 아니지만 우수한 성과를 올리고 있다. 우리나라에서도 SPR의 실천을 보다 적극적으로 추진할 필요가 있다고 본다.

6. 결 언

소프트웨어 프로세스를 체계적으로 개선하는 프레임워크를 SPR의 개념으로 정리하고 그 현상과 동향을 고찰하였다. 전술한 바와 같이 SPR의 기초인 프로세스의 개념은 BPR과 TQC와도 서로 통하는 것으로 새로운 개념은 아니지만 SPR은 다음의 2가지 점에서 소프트웨어 공학에 새로운 지평을 열 것으로 기대된다.

첫째, 프로세스의 형식적 표현방법의 제공:개인의 창조적 활동이라고 하는 개발활동의 배후에 있는 구조를 프로세스로서 형식적으로 표현한다.

둘째, 형식적으로 표현한 프로세스에 기초한 공학적이고 체계적인 방법론에 의한 개선이 가능하다.

또한 SPR의 연구와 개발에서는 소프트웨어 프로세스에 그치지 않고 BPR과 그룹웨어 등의 오피스 작업의 프로세스 기술, 하드웨어 제조의 프로세스 기술, 그리고 하드웨어에서 시작하여 소프트웨어에도 대응되고 있는 컨커런트공학 등의 관련기술로부터 얻을 수 있는 점이 많다고 본다. 앞으로의 연구과제는 이와 같은 기반기술을 활발히 하고 소프트웨어 프로세스의 설계, 개선의 방법을 소프트웨어 프로세스 공학(Software Process Engineering)으로서 체계화할 필요가 있다고 본다.

참 고 문 헌

1. Aoyama, M., "Distributed Concurrent Development of Software Systems:An Object-Oriented Process Model", Proc. IEEE COMPSAC '90, pp. 330-337, 1990.
2. Aoyama, M., "Concurrent Development Process Model", IEEE Software, Vol. 10. No. 4, pp. 46-55, 1993

3. Grady, R. B. and Slack, T. V., "Key Lessons in Achieving Widespread Inspection Use", IEEE Software, Vol. 11, No. 4, pp. 46-57, 1994.
4. Humphrey, W. S., "Managing the Software Process Improvement at Hughes Aircraft", IEEE Software, Vol. 8, No. 4, pp. 11-23, 1991.
5. Mi, P. and Scacchi, W., "Process Integration in CASE Environments", IEEE Software, Vol. 9, No. 2, pp. 45-53, 1992.
6. Osterweil, L., "Software Processes are Software Too", Proc. 9th ICSE, pp. 2-13, 1987.
7. Perry, D. E. et al., "People, Organizations, and Process Improvement", IEEE Software, Vol. 11, No. 4, pp. 36-45, 1994.
8. 青山幹雄他, "ソフトウエア開發のコストダウンモデルとその適用", 情報處理學會ソフトウエア工學研究會, No. 92-SE-84-5, 1992.
9. 青山幹雄, "People Meet Process:開發プロセスとの出會いと對峙", ソフトウエア技術者協會ソフトウエアシンポジウム '94 論文集, pp. 54-62, 1994.
10. 富士通通信ソフトウエア開發部(編):富士通における'あゆみ'活動, 日科技連, 1992.
11. 大野耐一, トヨタ生產方式, ダイヤモンド社, 1978.
12. 山本里枝子他, ソフトウエアプロセスに基づくソフトウエア開發環境の檢討, 情報處理學會'ソフトウエアプロセスシンポジウム'論文集, pp. 1-10, 1994.
13. 양해술, 이용근, 허태경, "소프트웨어 프로젝트 관리에서의 품질보증 시스템의 프로세스 기술 방식", 한국정보처리학회, 정보처리 Vol. 1, No. 3, 1994. 9.
14. 양해술, 김명옥, 박정호, "분산개발환경의 현상과 전망", 한국정보처리학회, 정보처리 Vol. 2, No. 1, 1995. 3.
15. 양해술, 이용근, 이하용, "프로세스 모델기반 개발 방법과 프로세스의 평가", 한국정보과학회, 정보과학회지 Vol. 13, No. 9, 1995. 9
16. 양해술, 최호득, 황인수, "소프트웨어 품질보증과 평가를 위한 ISO 900-3의 적용", 한국정보처리학회 추계학술발표논문집, 제2권 2호, 1995. 10

양 해 술

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)
1978년 성균관대학교 정보처리 학과 정보처리전공(석사)
1991년 日本 오사카대학교 기초공학부 대학원 정보공학과 소프트웨어공학 전공(공학박사)
1975년~79년 육군중앙경리단 전자계산실 시스템분석장교
1986년~87년 日本 오사카대학교 객원연구원
1993년~94년 한국정보과학회 학회지 편집부위원장
1980년~95년 강원대학교 전자계산학과 교수
1994년~95년 한국정보처리학회 논문지 편집위원장
1994년~현재 한국산업표준원(IIS) 이사
1995년~현재 한국소프트웨어품질연구소(INSQ) 소장
관심분야: 소프트웨어 공학(특히, S/W 품질보증과 품질평가, SA/SD, OOA/OOD/OOP, CASE, SI), 소프트웨어 프로젝트관리

이 용 근

1988년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1994년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1989년~92년 강원대학교 전자계산학과 조교
1995년~현재 강원대학교 대학원 전자계산학과 박사과정
1994년~현재 한림전문대학 전산정보처리학과 강사
관심분야: 소프트웨어공학(특히, S/W 품질보증과 품질평가, 객체지향 프로그래밍, 객체지향 분석과 설계 방법)

황 인 수

1978년 서울대학교 농과대학 농학과 졸업(학사)
1995년 정보처리기술사 자격 취득
1981년~88년 총무처 정부전자계산소 전산처리관
1989년~현재 삼성데이타시스템(주) 정보기술연구소 수석연구원
관심분야: 소프트웨어공학(프로젝트관리론, 품질관리, 시스템 분석과 설계, 시스템 통합)