□특집□

클라이언트/서버 개발의 품질관리 방법

양 해 술+

◈ 목 차 ◈

1. 서 론

오픈화, 다운사이징의 움직임에 호응하여 클라이언트/서버 시스템(CSS:Client/Server System) 개발의 요구가 증대하고 있다. 그러나 아직까지 클라이언트/서버 환경의 소프트웨어를 개발하기 위한 방법론에 대한 체계화된 표준은 없으며 여기에 대해서는 현재 지속적인 연구가 진행되고 있다. 클라이언트/서버 시스템 구축의 목적은 크게 다음과 같이 2가지로 분류할 수 있다.

첫째, 업무의 흐름을 수정하고 완전히 새로운 패러다임에 의해 시스템을 실현한다.

둘째, 시스템을 가능한한 신속하게 저비용으로 실현한다.

즉, 클라이언트/서버 시스템 개발의 프로세스는 종래의 시스템 개발의 프로세스와 전혀 다른 프로세스라는 것이 여러 가지 사례 분석을 통하여 입증되었다. 또한, 위의 두 가지 목적에 대응한 2개의 계층으로부터 만들어진 반복확대형의 개발 프로세스에 기초한 새로운 클라이언트/서버 시스템 개발 방법론에 제안되고 있다.

본 고에서는 먼저 클라이언트/서버 개발 방법에 대해 살펴본 후에 제 3장에서는 클라이언트/서버 개발에서 특히 중요한 품질특성에 대해 서술하였다. 끝으로, 제 4장과 5장에서는 클라이언트/서버 개발에서의 품질관리와 분산 병행개발의 품질관리 방법에 대해 기술하기로 한다.

2. 클라이언트/서버 기반 개발 방법

2.1 클라이언트/서버 시스템의 개발

종래의 메인프레임 소프트웨어 개발에서 폭포수형 수명주기의 각 공정 작업 내용과 필요한 요원의 기술 레벨에 관하여 개발 기관과 발주기관에 대한 공통 인식이 확립되어 있다. 즉, 메인프레임 소프트웨어 개발과 달리 클라이언트/서버 환경에서 요구되는 기술들이 종래와는 다르다는 것이며 필요한 기술들의 예를 들면 다음과 같다.

+ 종신회원 : 한국소프트웨어품질연구소(INSQ) 소장

- PC나 WS(Workstation)의 하드웨어와 운영체제
- LAN 관련 기술
- 호스트 컴퓨터와 클라이언트/서버 환경과의 접속
- 클라이언트/서버 환경에서 활용가능한 개발 도구
- 윈도우 시스템
- 그래픽 유저 인터페이스 개발
- 프로토타이핑 개발과 평가
- 클라이언트/서버 환경에 적합한 모델링 기법
- 데이타 보관장소와 프로세스 처리장소와의 최적 배치

또한, 클라이언트/서버 시스템 개발부문의 핵심은 지금까지 메인프레임 소프트웨어 개발을 중심으로 축적해 온 기술력이 클라이언트/서버 시스템의 개발에서는 활용되지 못한다는 점이다. 워크스테이션(WS)이나 PC에 의한 소프트웨어 개발이 메인프레임의 소프트웨어 개발과정에서 사용해 온 종래의 기법과 다른 점으로 종래의 기법만으로는 우수한 개발자가 실패할 확률이 높다는 것을 들 수 있다. 그 차이점을 비교하면 <표 1>과 같다.

<표 1> 클라이언트/서버 SW 개발과 메인프레임 SW 개발의 차이점

구분	클라이언트/서버 소프트웨어	메인프레임 소프트웨어
설계 시점	• 사건에 대하여 프로세스가 발생하는 방식에서 설계	• 데이타베이스 처리를 중심으로 프로세스를 설계
개발 프로세스	• 설계와 개발이 분리되어 있지 않음 • 명세서없는 개발도 있음	• 설계와 개발이 분리되어 있음 • 명세서를 작성하는 것이 원칙
필요한 지식	• 하드웨어에서 OS, DB, 네트워크, 개발지원 도구에 관한 폭넓은 지식	• 중간 소프트웨어가 준비되어 있지 않기 때문에 상세한 지식은 불필요

그러나, 이와 같은 차이점을 전부 습득하고 있는 기술자는 아직 많지 않으며 각각의 기술에 있어서도 그것을 어떤 체계로 활용할 것인가에 대한 확립된 공통 의식이 존재하지 않는다. 따라서, 현재 클라이언트/서버 환경에서의 소프트웨어 개발은 소규모 개발 대상에서 점차 대규모 대상에 대하여 독자적인 개발 기법으로 실시하여 통합해 가는 실정이다.

(1) 클라이언트/서버 시스템의 개발자 요구

클라이언트/서버 시스템 개발부문에서 필요한 부분은 <표 2>와 같이 프로젝트의 멤버는 클라이언트/서버 시스템 개발의 각 부분에서 하드웨어로부터 소프트웨어까지 다양한 사상을 거의 동시에 고려하여 지원을 요구하고 있다는 점이다.

<표 2> 클라이언트/서버의 개발자 요구 지원환경

분 류	요 구
데이타 베이스	• 점유율이 높은 관계 데이타베이스를 전부 취급하는 간결한 방법으로 데이타베이스 취급 • 상세한 데이타베이스 제어와 처리도 실행
네트워크	• 점유율이 높은 네트워크 OS를 사용한 시스템을 쉽게 구축
유저 인터페이스	• Motif, Windows 등의 화면 개발 용이 • 가능하면 동일한 방식으로 다양한 그래픽 유저 인터페이스를 취급
전체	• COBOL의 10배 이상의 생산성을 얻을 수 있음

(2) 클라이언트/서버 개발의 프로젝트관리 특징

클라이언트/서버 시스템 개발 프로젝트의 주된 특징은 다음과 같은 일반적인 경향이 명확해짐을 알 수 있다.

- 프로젝트 규모가 점차 대규모화되고 소수 정예의 개발팀이 서로 통합하여 개발이 진행되고 있다.
- 하드웨어로부터 어플리케이션까지 폭넓은 지식, 기술을 확보한 유능한 리더, 톱매니저나 사용자를 설득시키는 CIO와 같은 리더가 필요하다.

2.2 클라이언트/서버 시스템의 BPR적 측면

클라이언트/서버 시스템 개발의 요건은 사용자의 만족도가 높은 시스템을 신속히 개발하는 것이다. Michael Hammer가 제창한 BPR(Business Process Reengineering)의 사고방식에서 클라이언트/서버 시스템의 프로세스를 만족하기 위해서는 전문화된 개발팀이 순서적으로 분담 업무를 완수하는 형태가 아닌, 한사람, 또는 하나의 팀이 일괄적으로 프로젝트를 실현하고 사용자 환경에 시스템을 조기에 적용시켜야 한다.

한편, 컴퓨터 시스템에 관련된 기술의 진보가 빠르지만 운영체제, 미들 소프트웨어 등의 각 계층의표준화는 곤란하다. 그리고 각각의 업무 담당부서가 역할을 분담하여 시스템 개발을 하는 경우에는 그 변화에 대응할 수 없게 된다.

이와 같은 사고방식에서 클라이언트/서버 시스템 개발이라는 업무에 대해 프로세스의 근본적인 수정, 바꾸어 말하면 BPR이 절대적으로 필요하다는 결론에 도달한다.

적절한 프로세스로서 권장하는 개발 방법은 2계층이며 각 계층이 반복확대형(나선형 모델)의 개발 프로세스이다. 다음에 본 프로세스와 그것에 기초하는 클라이언트/서버 시스템 개발 방법론에 대해 기술한다.

2.3 반복확대형 모델에 기초한 개발 방법

(1) 2계층 프로세스

클라이언트/서버 시스템 구축에서는 다음과 같은 2가지 목적이 있다.

첫째, 새로운 패러다임의 시스템을 구축하는 것보다 작업의 흐름을 효율적으로 변경한다.

둘째, 시스템을 적은 비용으로 신속히 개발하는 것이다.

이와 같은 이질적인 목적에 유연하게 대처하기 위해서는 2계층으로 분산하여 상위 계층은 가치있는 클라이언트/서버 시스템을 제공하는 것을 목적으로 작업의 분석, 신업무의 요구분석으로 시작하고 새로운 플랫폼에서의 시스템 구축 실현성의 확인까지를 행한다. 구체적으로는 클라이언트/서버 시스템 구축의 목적을 명확히 하고 그 목적에 부합한 업무, 정보의 분산 방법을 검토하여 업무에 상당히 깊이 관련되기 때문에 사용자의 적극적인 참여가 필요하다.

하위 계층은 클라이언트/서버 시스템을 신속하게 저가로 실현하는 부분으로서 여기에서 중요한 것은 명세서를 조기에 확정하고 불안 요인 특히, 성능 문제를 조기에 배제하는 것이라고 할 수 있다. 이것을 실현하기 위해서는 프로세스도 물론 중요하지만, 이용하는 개발지원 도구를 선택하는 방법도 크게 영향을 미치므로 좋은 도구를 제대로 사용하는 것이 성공의 열쇠가 된다.

(2) 반복 확대형 프로세스

클라이언트/서버 시스템의 두가지 목적을 실현하기 위해 프로세스에 대해 생각해 보기로 한다. 먼저 기술한 2계층의 각각에 대해 유효한 프로세스란 가치있는 클라이언트/서버 시스템을 계획하는 계층이지만 여기에서는 클라이언트/서버 시스템 구축의 목적을 명확히 하고 그 목적에 적합한 업무와 정보의 분산을 검토해야 한다. 그러기 위해서는 어떠한 가치를 만들고 싶은가를 명확히 하고 그것을 실현할 때의 업무, 조직, 체제, 정보의 특성에 의한 다양한 제약 조건을 총합적으로

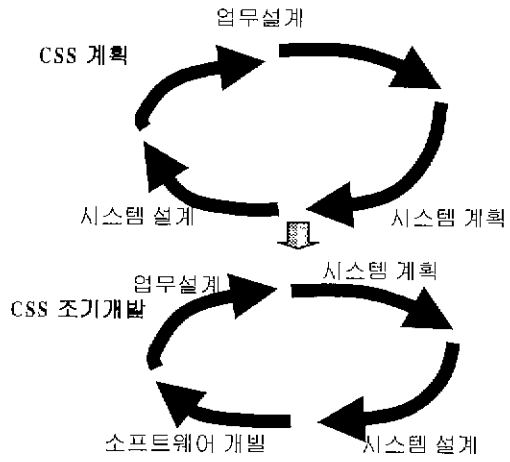
고려해야 하며 업무와 정보의 분산화 기준을 설정할 필요가 있다. 그리고 그 기준으로 삼아 업무와 정보를 배치해 보고 동시에 어느정도 레벨까지 클라이언트/서버 시스템의 구성을 설계하고 그 실현 가능성을 검토해 둔다.

이 일련의 프로세스는 사용자와 개발자가 협력하여 실행해야 하며 이 프로세스는 1회에서 마무리되는 것이 아니고 몇번이고 반복함으로써 검토 내용과 목적이 명확해지며 보다 실현성이 높은 계획이 된다.

다음에 클라이언트/서버 시스템을 조기에 적은 비용으로 개발하는 계층에 대해 생각해 보기로 한다. 이것은 2가지 관점으로 생각할 수 있으며 첫 번째는 미확정된 부분을 조기에 확정하는 것이고, 두 번째는 우선적으로 필요한 최소의 기능만 실현하여 사용자에게 제공하고 나중에 확장시켜 나가는 것이다.

전자의 프로세스는 우선 시스템의 명세를 사용자에게 가시화된 형태로 제공하고 사용자와의 생각 차이를 명확히 하여 명세를 확정해가는 것과 시스템을 구축할 때의 불안 요인, 예를 들면 신뢰성이나 성능 문제 등을 조기에 배제한다는 점을 들 수 있다. 결국 사용자와 함께 적용 검토를 반복하면서 진행하는 방법이다. 후자의 경우에도 핵심이 되는 기능을 실현하면 그것을 적용하면서 기능을 추가해 나가는 반복의 프로세스가 된다. 이와 같이 2계층은 유사한 프로세스를 채택해야 하며 2계층 반복확대형 프로세스의 개요는 (그림 1)과 같다.

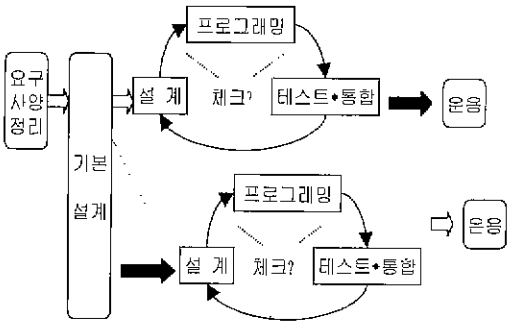
상위 계층인 클라이언트/서버 시스템 계획은 가치있는 클라이언트/서버 시스템을 계획하는 프로세스이고 업무분석으로부터 클라이언트/서버 시스템의 구성 설계까지를 한다. 필요하면 성능 설계, 신뢰성 설계까지 행하고 시스템의 실현성을 확인한다.



(그림 1) 2계층 반복 확대형 프로세스

하위 계층인 클라이언트/서버 시스템 조기 발달은 클라이언트/서버 시스템을 조기에 저가로 개발하는 프로세스이고 명세의 조기 확정과 성능의 조기 견적이 포인트라고 볼 수 있다.

이상과 같은 2계층 프로세스를 채택한 클라이언트/서버 개발 모델의 예는 (그림 2)와 같다. (그림 2)와 같은 개발 모델에서는 10인 정도의 소규모 팀이 병행하여 개발 작업을 진행하며 팀내의 개발 스타일은 메인 프레임 시대와 같은 형태의 접근 방법을 채택할 수 있다.



(그림 2) 클라이언트/서버 환경에서의 개발 모델의 예

위와 같은 클라이언트/서버 환경에서 많이 사용되고 있는 개발에서는 데이타 모델 등의 기본 설계후에 시스템을 분할하여 복수의 소수팀이 스파이럴 어프러치(Spiral Approach)로서 병행 개발을 진행하고 있다. 이와 같은 모델의 특징은 생산성을 향상시킬 수는 있지만 품질은 개발자의 능력에 좌우되는 단점이 있다.

3. 클라이언트/서버 개발의 품질 특성

3.1 클라이언트/서버 개발의 품질관리 문제점

클라이언트/서버 개발에서 품질관리의 어려움을 해결하기 위해서는 먼저 종래의 개발과 비교했을 때, 클라이언트/서버 개발에서 변화된 점과 동일한 점을 명확히 인식할 필요가 있다. 그 다음에 품질관리의 기본 개념을 이해할 수 있으면 클라이언트/서버 개발에서의 품질관리도 응용문제로서 해결할 수 있다. 일반적으로 클라이언트/서버 지향 소프트웨어 개발에서 고품질을 유지하기 어려운 원인은 다음과 같은 과제들이 있기 때문이다.

(1) 기본 소프트웨어의 불안정

이 문제는 플랫폼이 불안정하다는 것으로서, 메인프레임에서는 메이커가 플랫폼의 신뢰성을 책임지고 있지만 클라이언트/서버 시스템에서는 OS나 데이타베이스 등을 조합할 때 안정적인 기동에 대해 사용자나 SI 업체가 책임져야 하기 때문에 발생한다. 즉, 메인프레임에 비해 클라이언트/서버 시스템에서는 개발자 개인의 능력에 따라 시스템의 성능이 크게 달라지게 된다.

(2) 개발 스타일의 변화

클라이언트/서버 시스템에서의 두 번째 문제점은 소프트웨어 개발 스타일이 다르다는 것이다. 즉, 메인프레임 시대의 개발은 요구사양의 분석에서부터 설계, 구현, 테스트, 운용의 각 단계를 순서대로 진행해가는 폭포수형 모델 중심이었다. 그러나 클라이언트/서버 환경에서는 복수의 소수팀이 스파이럴 접근기법 즉, 설계, 프로그래밍, 테스트를 회전형태로 몇번이고 반복해가면서 시스템을 병행 개발하는 패턴으로 대규모 시스템을 개발하고 있다. 이와 같은 새로운 개발 스타일에 대응하기 위한 품질관리 방법은 아직 확립되지 못하고 있다.

(3) 테스트 곤란한 Visual 툴

클라이언트/서버 시스템에서는 클라이언트 측의 소프트웨어를 개발하기 위해 Visual 개발 툴을 사용하는 경우가 많다. 즉, Visual 개발 툴로 개발한 시스템은 마우스의 클릭이나 Key 입력의 이벤트가 발생한 시점에서 처리를 하는 이벤트 구동형 프로그램이다. 따라서, 유저 인터페이스를 쉽게 하기 위해 메뉴나 버튼이 많이 사용되고 화면 이동 패턴의 증가로 기능의 조합이 방대하기 때문에 충분한 테스트를 할 수 없게 된다.

3.2 클라이언트/서버 개발환경의 이점

클라이언트/서버 개발환경은 각 개발자의 레벨로부터 개발 환경 전체까지 다음과 같은 개선효과를 가지고 있다.

(1) 사용자의 유용한 환경

고기능 PC 또는 워크스테이션의 강력한 처리능력을 활용하여 각 개발자의 작업환경을 개선할 수 있다. 즉, 응답시간의 개선, GUI에 의한 정보 표현력의 향상, 네트워크에 의한 정보의 수집과 교환의 개선, 조직분산 등과 같이 종래와 다른 개발 패러다임에 의한 개발 환경 개선을 목표로 하고 있다.

(2) 개방 환경

클라이언트/서버 개발의 개방 환경은 개발 규모의 증대에 대응하여 개발 환경 구축을 위한 도입 비용이나 운용 비용이 절감될 수 있도록 확장이나 분산이 가능해짐에 따라 사용자에게 많은 이점을 제공하며, 아울러 다음과 같은 특성을 만족시킨다.

- 호환성(compatibility):국제표준 등에 준거하여 달라지는 구현(implementation)이나 버전 수간에 OS 인터페이스가 보증되고, 이용자의 선택의 폭이 넓어지고 기존 소프트웨어 자산이나 데이타가 활용될 수 있다.
- 이식성(portability):다른 시스템이나 OS간에 어플리케이션의 이식성이 보증되고 기존의 어플리케이션 자산이 활용될 수 있다. 예를 들어 다른 아키텍쳐를 가지고 있는 시스템 사이의 ABI(Application Binary Interface)가 있다.
- 확장성(scalability):동일한 어플리케이션이 워크스테이션으로부터 수퍼 컴퓨터에 이르기까지 다른 규모(scale)의 시스템 상에서 동작할 수 있게 된다.
- 상호운용성(interoperability):다른 아키텍처의 시스템간에도 네트워크로 상호접속이 이루어지고 통신이나 협조처리 등이 가능하다.

(3) 신뢰할 수 있는 환경

개발 조직이나 각종 서비스의 기능과 부하가 분산되므로 특정한 서버에 장해가 발생하더라도 클라이언트/서버 환경 전체가 정지할 위험이 분산됨으로써 개발작업 전체가 중단되는 일을 사전에 방지할 수 있다.

3.3 클라이언트/서버 개발의 품질 향상

클라이언트/서버 시스템의 품질을 유지하기 위한 최대의 난제는 개발 플랫폼의 불안정을 안정화시키는 작업에 달려있다고 해도 과언이 아니다. 이를 해결하기 위한 방안으로 대형 SI 업체가 실천하고 있는 대책의 주류는 다음과 같이 개발자의 자유를 제한하는 것이다.

- 독자적인 미들웨어를 도입하여 이용가능한 기능을 한정한다.
- OS나 트랜잭션 처리 모니터, 데이타베이스를 조합하여 작업 패턴을 한정한다.
- 클라이언트/서버 개발의 노하우를 축적하고 참조한다.
- 프로토타입을 구축하여 정상 동작을 사전에 확인한다.

이와 같이 개발자의 자유를 제한하는 패턴으로서 첫 번째 방법은 독자적으로 미들웨어를 도입하는 방법이 있다. 즉, OS나 트랜잭션 처리 모니터 등의 플랫폼상에 독자적으로 개발한 미들웨어를 배치하여 개발자가 직접 포착하지 못하게 하는 것이다. 이 방법은 80년대 말에 클라이언트/서버 시스템 개발시에 채용되기 시작한 방법이지만 유효성을 입증시키기 위해 많은 SI 업체가 전형적인 수단으로 채용하고 있다.

두 번째 패턴은 플랫폼의 조합을 한정하는 방법으로 시스템의 목적별로 표준 작업 패턴을 제공하여 품질향상을 하는 것으로서 시스템 통합에서는 표준작업 패턴 중에서 사용자에게 플랫폼을 제한하는 것이다.

세 번째의 노하우 축적 방법은 지금까지의 개발 경험을 축적한 노하우를 그룹웨어상의 전자게시판에 등록하고 개발자가 참조하는 방법이다. 이를 위해 대형 SI 업체에서는 사내에 개설한 오픈 솔루션 센터 등을 이용하고 있다.

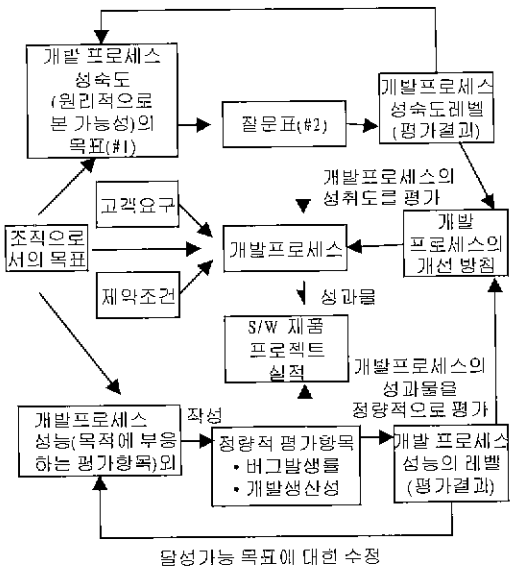
끝으로 클라이언트/서버 개발에서는 가능한 범위내에서 항상 프로토타입을 구축하여 사용자로 하여금 정상적인 적용이 가능하도록 지원하여야 한다.

4. 클라이언트/서버 개발의 품질 관리

클라이언트/서버 개발에서의 전체적인 품질에 관해서는 소프트웨어의 프로세스에 대한 평가와 프로세스로부터 생성된 성과물의 관련을 조사하여 프로세스의 개선을 성과물의 개선과 결부시키

는 연구가 활발히 전개되고 있다. 여기에서 말하는 프로세스는 소프트웨어의 생명주기 전반에 발생하는 작업을 총칭하는 것이며 프로세스를 나누는 방법은 ISO/IEC 12207의 "Software Life Cycle Process(SLCP)"가 국제 표준으로 규정하고 있다. 그리고 프로세스가 정확히 실시되고 있는가에 대한 평가 척도는 카네기 메론 대학의 생산기술연구소(SEI)에서 제안한 CMM(능력성숙도모델;Capability Maturity Model)이 적용되고 있으며 품질보증 프로세스를 규정한 것으로 ISO 9000 시리즈가 널리 이용되고 있다.

그러나 CMM이나 ISO 9000이 평가하는 것은 프로세스 자체에 대한 평가로서 프로세스가 만들어내는 소프트웨어의 품질이나 개발 생산성과는 거리가 멀다고 할 수 있다. 따라서 이와 같은 문제점을 해결하기 위해 품질관리 기법을 포함한 소프트웨어 개발 프로세스를 개선하기 위해 채용하고 있는 방법은 (그림 3)과 같다.



㈜ #1 : ISO/IEC 12207 SLCP, #2 : CMM 참조
#3 : ISO/IEC 9126 사용

(그림 3) 소프트웨어 개발 프로세스의 개선 방법

현재, 이른적으로 상기 그림과 같은 클라이언트/서버 환경 지향의 품질시스템을 도출하려는 접근방법이 활성화되지 않고 있으나 ISO 9000이나 CMM을 활용한 프로세스 개선에 몰두하면서 실제 개발경험을 축적하고 클라이언트/서버 시스템에 적합한 최적의 품질시스템을 모색하고 있다.

5. 분산 병행 개발의 품질관리

클라이언트/서버 개발 기간의 단축과 급속한 소프트웨어 성장에 대응하기 위해서 소프트웨어 분산 병행 개발이 중요시되고 있다. 소프트웨어의 분산 병행 개발이란 공간적으로 분산된 복수의 개발 거점에서 시간적으로 병행해서 소프트웨어 개발을 처리하는 개발 패러다임이라고 할 수 있다.

5.1 분산 병행 개발에 의한 품질관리의 문제

클라이언트/서버 개발에서 주로 많이 이용되고 있는 분산 병행 개발은 새로운 개발 형태이기 때문에 그 품질관리에 대한 새로운 문제에 직면하게 되었다.

첫째, 분산 개발에 의한 본질적인 문제는 정보의 생성, 보존, 관리가 분산되어 있다는 것이다. 예를 들면, 품질의 척도는 거점간에 서로 다른 경향이 있고, 품질 정보가 분산되기 때문에 그 수집과 인식 및 개발 프로세스로의 피드백에 시간이 걸리고 비용도 많이 소요된다. 이로 인해 최신의 정보를 정확히 파악하는 것이 곤란해지고, 거점간에 정보의 차이가 발생하는 경향이 있다.

둘째, 병행 개발에 의한 본질적인 문제는 개발 작업과 프로덕트의 변화가 병행해서 생기는 것에 의한 변경과 병합이다. 예를 들어 동일 모듈을 복수의 개발자가 변경하는 경우 병합에 의한 내용의 일관성이 상실되는 경우가 발생한다.

따라서 분산 병행 개발의 품질관리를 처리하는

데는 개발 프로세스와 프로덕트의 품질간에 관계하는 정보를 정확히 파악하고, 시간적으로나 공간적으로도 차이가 없는 관리를 실현하는 것이 필요하다.

5.2 분산 병행 개발의 품질정보 구조

분산 병행 개발의 품질관리에 있어서는 분산된 품질 정보의 생성, 보수, 교환 등을 처리하는 것이 필요하다. 이를 위해서는 분산 처리 시스템의 설계와 같은 분산 병행 개발에서의 품질 정보 모델을 명확히 할 필요가 있다. 분산 병행 개발에 관한 품질 정보의 흐름에 착안한 품질관리 모델은 어플리케이션 개발에 관한 구조화 분석 기법을 이용하였다. 이와 같은 기법에 의한 품질 정보의 모델을 품질정보 구조라 부르고, 품질 정보 구조에 기초하여 개발 프로세스에 따르는 품질 정보의 흐름과 그 변화를 음미함으로써 분산 병행 개발에 관한 품질 관리 모델화를 처리할 수 있다.

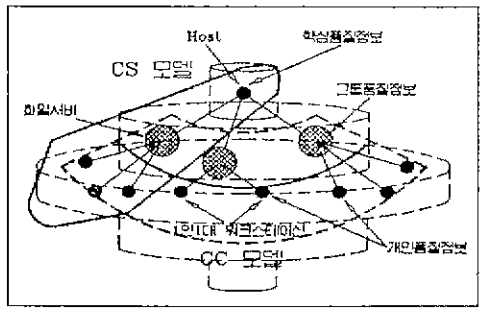
분산 병행 개발에 관한 품질정보 아키텍처는 (그림 4)와 같이 다음의 두가지 구조적인 클래스로 분류할 수 있다.

- 수직분산형 품질정보 구조 : 대상 시스템과 개발조직의 계층구조를 반영한 계층적 정보 구조로서, 그룹마다의 품질정보 및 개발 거점, 프로젝트 전체의 품질정보를 나타낸다.
- 수평분산형 품질정보 구조 : 동일 수준에 있는 분산 개발거점과 그룹사이에서 분산된 동일 구조를 갖는 품질 정보로서 각 그룹과 모듈마다의 버그 발생 정보를 나타낸다.

5.3 품질 관리의 지원 환경 모델

분산 병행 개발에 관한 2개 클래스의 품질정보 흐름을 원활히 지원하기 위해서 (그림 4)에 표시한 품질관리 지원 환경 모델은 다음 2개의 분산처리 모델을 통합한 품질관리를 위한 분산 개발 환경이라고 할 수 있다.

- CS(Client Sever) 모델 : 수직 분산형 품질정보 구조를 지원하는 환경모델로서 WS와 서버가 연계하여 처리한다. 1인 1대의 WS는 클라이언트로서 개인 또는 그룹 단위 품질정보의 입력을 표시하고 있으며, 클라이언트의 요구에 따라 호스트 상의 대규모 DB를 액세스하거나 데이타를 수집하는 것 등은 서버가 처리한다. 이처럼 CS 모델에서는 클라이언트와 서버가 자연스럽게 계층적으로 역할을 분담하고 있다.
- CC(Co-operative Computing) 모델 : CC 모델은 수평분산형 품질정보 구조를 지원하는 환경 모델로서, 수평분산된 1인 1대의 WS가 협조하여 병행하면서 처리를 진행하고 있다.



(그림 4) 분산 병행 개발과 품질정보 구조

CS 모델과 CC 모델을 통합한 품질관리 지원 환경은 계층적인 구조를 가짐으로써 조직의 규모와 형태가 다른 경우에도 유연하게 대응할 수 있다. CS 모델은 프로젝트 전체의 계층구조를 반영한 전역적인(global) 품질정보 관리 기구이며, CC 모델은 그룹 거점내에서 정보를 생성하거나 교환하기 위한 국소적인(local) 품질정보 관리 기구이다.

6. 결 론

최근, PC나 워크스테이션의 성능이 획기적으로 향상되고 LAN관련 기술 및 엔드 유저의 사용 편리성을 최대화하는 윈도우 시스템 같은 그래픽 사용자 인터페이스 기술이 발전함에 따라 오픈화, 다운사이징 등으로 컴퓨팅 환경이 빠르게 변화해 가고 있다. 이에 따라 S/W 개발에 대한 요구 또한 클라이언트/서버 시스템을 구축하는 방향으로 전환되어 가고 있는 추세이다.

기존의 메인프레임 중심의 정보시스템이나 개발 환경에 비해 PC나 워크스테이션 중심의 환경이 가지고 있는 다양한 장점들 때문에 지금까지 클라이언트/서버 기반의 개발 방법에 대한 연구가 활발히 이루어져 왔으며 클라이언트/서버 시스템이 기업 정보시스템의 핵심을 담당하고 있으나 클라이언트/서버 시스템을 개발하는 과정에서 품질관리 방법 측면의 많은 문제점이 나타나고 있다. 예를 들어, 클라이언트/서버 개발 플랫폼이 불안정한 문제라든가 개발 스타일의 변화에 미처 대처하지 못한 점, 클라이언트/서버 시스템을 Visual 툴을 이용하여 개발했을 때 충분한 테스트가 곤란한 점 등이다.

그러나 이러한 문제점으로 인해 발생가능한 시스템 다운 등의 문제를 사전에 방지하기 위한 품질관리 기법이 확립되어 있지 않다. 이와 같은 점을 고려하여 지금까지 먼저 2계층 반복확대형 프로세스에 준하여 클라이언트/서버 시스템 개발 방법론에 대하여 기술하였다. 그리고 클라이언트/서버 개발에서 필요한 품질의 특성과 효과적인 품질관리를 위한 프로세스의 개선과 평가에 대해 기술하였으며 끝으로 클라이언트/서버 개발에서 많이 채택하고 있는 분산 개발의 품질관리 방안에 대해 살펴 보았다. 이와 같은 방법은 앞으로 개발의 운용, 보수까지 포함한 프로세스로 구축해 갈 예정이다. 또한, 프로젝트 관리 방법, 품질관리 방법, 견적, 계약 방법에 대해서도 실 프로젝트에서의 적용사례를 분석하는 것에 의해 명확히 그 체계를 정립할 필요가 있다.

참고문헌

[1] Hammer, M., "Reeengineering Work", Harvard Business Review, Aug./Sep., 1990.

[2] Champine, G. A., Geer, Jr., D. E. and Rug, W. N., "Project Athena as a Distributed Computer System", IEEE Compter, Vol. 24, No. 9, pp. 40-51, 1990.

[3] Harrison, W. H., Ossher, H. and Sweeney, P. F., "Coordinationg Concurrent Development", Proc. ACM CSCW'90, pp. 157-168, 1990.

[4] Grief, I. (ed,), "Computer Supported Cooperative Work", Morgan Kaufmann, 1988.

[5] Alex Berson, "Client/Server Architecture", Mc-Graw-Hill, pp. 36~383, 1992.

[6] J. J. Shedltsky, "Application reference designs for distributed systems", IBM Systems Journal, Vol. 32, No. 4, pp. 625-646, 1993.

[7] Jerry Cashin, "Client/Server Technology, The New Direction in Computer Networking", Computer Technology Research Corp., 1992.

[8] 青山, 小池, "交換ソフトウェアの分散並行開發支援環境", 電子情報通信學會 交換システム研究會, No. SSE90-25, pp.7-12, 1990.

[9] 石井, "グループウェアの研究動向", 情報處理, Vol.30, No. 12, pp.1502-1508, 1989. 12.

[10] 양해술, 이용근, 이하용, "프로세스 모델기반 개발 방법의 프로세스의 평가", 한국정보과학회지 제 13권 제9호, 1995. 9.

[11] 양해술, 김명옥, 박정호, "분산 개발환경의

현상과 전망", 한국정보처리학회, 정보처리학회지, Vol. 2, No. 1, 1995. 3.

[12] 양해술, 안유환, "소프트웨어 분산 병행개발의 품질관리 방법", 한국정보처리학회, 춘계학술발표논문집, 제4권, 제1호, 1997. 4.

양 해 술

1975년 홍익대학교 공과대학 전기공학과 (학사)
1978년 성균관대학교 정보처리학과 정보처리 (석사)
1991년 日本 오사카대학교 기초공학부 정보공학과 소프트웨어공학 (공학박사)
1975년-79년 육군중앙경리단 전자계산실 시스템분석장교
1886년-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