□ 기획연재 □

컴퓨터 과학 산책(13)

로마인과 만물박사

서울대학교 우치수*

천 수 백년 전에 로마에 살았던 사람의 이야기인 일본 여류 작가 시오노 나나미의 '로마인 이야기'란 책이 요즘 베스트셀러 반열에 올라선 모양이다. 시간적으로나 지리적으로 멀리 떨어진 로마에 대한 기록이지만 한 나라나 조직의 흥망성쇠에 대한 얘기는 흥미로울 뿐만 아니라 시사해 주는 바가 크다.

작가는 독자에게 이런 질물은 던진다. 로마 제국의 힘의 근원은 무엇일까? 고도의 문화를 이룩한 그리스는 왜 그렇게 빨리 몰락한 반면 로마의 융성은 왜 그렇게 오래 지속되었을까? 여러 사료에서 볼 수 있듯이 로마인은 지성에서는 그리스인보다 못하고, 체력에서는 켈트인이나 게르만인보다 못하고, 기술력에서는 에트루리아인보다 못하고, 경제력에서는 카르타고인보다 뒤떨어진다고 한다. 특별히 내세울만한 장점이 없는 민족이 저 광대한 영역에 걸친 대제국을 그토록 오랫동안 경영할 수 있었던 이유는 무엇일까?

쉽게 답을 내릴 수 없는 질문이지만 작가는 유일신을 고집하지 않는 그들의 종교관과 모든 계급의 힘을 결속시킬 수 있었던 그들의 사회제도, 상대를 포용하여 자신에게 동화시켜 버린 그들의 개방적 성향을 이유로 내세우고 있다. 왕정의 장점을 살린 집정관 제도, 귀족정의 장점을 살린 원로원 제도, 민주정의 장점을 살린 민회로 구성된 로마 공화정의 독특한 정치체제는 사회의 힘을 하나로 결집시켜 귀족과 평민으로 구성된 강력한 군사력을 가능케한 원동력이 되었다는 것이다.

결국 한 조직의 흥망성쇠에 있어서 기술력이나 경제력이 물론 중요한 요인으로 작용하지만 그보다도 조직의 구성이나 제도와 같은 관리적 측면이 다른 어떤 것보다도 중요하다는 결론을 내린다면 비약이 너무 심한 것일까?

다른 공학 분야와 마찬가지로 소프트웨어 공학은 유도된 응용 분야이다. 기초 원리나 이론은 전산, 수학, 심리학, 경영학, 경제학 등 좀 더 기초적인 분야의 연구를 이용한다. 소프트웨어 공학은 다른 분야보다 관리적 측면이 더 강하다고 말 할 수 있다. 소프트웨어 공학자는 다른 분야의 기술과 지식을 이용하여 좋은 품질의 소프트웨어를 효과적인 방법으로 생산할 수 있도록 관리하는 것이 목표이다.

하드웨어의 급속한 발전과 더불어 소프트웨어 요구가 증대되면서 소프트웨어 수요는 공급능력을 초과하였다. 갈수록 심해지는 공급과 수요의 격차를 줄이기 위해 1970년대에 소프트웨어 공학이 탄생하여 지금까지 많은 기법이 개발되고 발전되었으나 근본적인 해결책은 아직 없으며 공급과 수요의 격차는 더 심화되는 추세를 보이고 있다. 갈수록 더 크고 복잡해져 가는 소프트웨어 시스템에 대처하기 위해 1990년대 초에 도입된 객체 지향 기법은 현재 계속 발전하는 단계이며 모든 문제를 해결하기에는 아직 부족하다.

결국 근본적인 해결책은 '소프트웨어 공장'에서 '소프트웨어 부품'의 조립으로 필요한 소프트웨어 시스템을 자동 생산할 수 있는 기법과 도구일 것이다. 그 기법과 도구는 부품으로 구성된 거대한 규모의 분산 시스템, 필요에 따라 필요한 기능을 갖는 부품을 자유롭게 교체 추가 할 수 있으며 부품의 추가만으로 더 크고

*종신회원

더 복잡한 문제도 해결할 수 있는 시스템을 가능케 해야 할 것이다. 가깝게 내다보면 객체 지향 기법이 이에 대한 해답을 줄 것으로 기대되고 현재 많은 연구가 진행되고 있으나 아직 선결되어야 할 문제가 많이 남아 있는 실정이다.

소프트웨어 부품이란 측면은 객체 지향 기법의 주요 장점으로 많이 언급되고 있으나 현재 성과는 기대에 크게 못 미치고 있다. 일반적으로 사용되기 위한 인터페이스 구조, 부품 개발 기법, 부품에 대한 지적 자산권 소유권 문제, 부품으로 개발된 시스템의 평가 문제 등 먼저 해결되어야 할 사항이 아직 많이 남아있다.

많은 수의 부품으로 구성된 거대한 시스템의 설계에서 기존의 방법은 부적당하다. 불과 몇 년 안에 요청될 몇 십만 혹은 몇 백만개의 부품으로 구성된 거대한 시스템의 경우 그 구조와 이들 부품들 간의 상호 작용 설계 문제에 있어서 아직 이렇다할 해답은 없다. 이렇게 많은 수의 부품으로 구성된 시스템에 있어서 자원 관리 하부구조는 어떻게 설계되어야 하는가? 성능 예측이나 시스템 평가를 위한 모델은 어떤 형태이어야 하는가? OMG 그룹에서 개발된 CORBA는 큰 규모의 분산 객체 시스템에 대한 중요한 첫 발전이지만 아직 이런 문제에 대한 해결책을 제시해 주고 있지 못하다.

개발 프로젝트 측면에서도 크기는 문제가 된다. 거대한 시스템의 개발 프로젝트는 많은 인원과 시간과 자원을 소요할 것이다. 기존의 프로젝트 관리 기법이 이처럼 거대한 프로젝트의 경우에도 잘 적용될 수 있을 것인가?

정보 시스템은 몇 년 안에 급격히 변화할 것이다. 앞으로는 유산 코드(legacy code)뿐만 아니라 유산 데이타(legacy data)까지 걱정해야 할 것이고 유산 데이타는 데이타 웨어 하우스(data warehose) 형태의 관리가 요구될 것이다. 분산 처리, 멀티미디어, 인트라넷, 글로벌라이제이션(globalization)은 이제 일반적인 용어가 되어가고 있다. 여기에 발 맞추어 소프트웨어 시스템 개발과 관련된 기법, 도구, 기술 역시 급속한 속도로 변화하고 있다.

개발 방법론은 통합 표준화 되어가는 방향으로 나아가고 있다. 객체 지향 방법론의 표기법 통합이 시도되고 있고 여러 방법론의 특징을 융합 개선시킨 혼합형 방법론이 많이 사용되고 있다. 모든 응용 분야에 적합한 방법론은 없을 것이므로 큰 시스템일수록 여러 측면에서 각각 적합한 방법론을 사용하여 개발될 것으로 예상된다. 이렇게 사용되는 여러 방법론은 CASE (computer aided software engineering) 측면에서 통합되어야 할 것이다.

CASE 도구는 개발 방법론과 성공적으로 결합되어 적용되고 있고 분산 처리 기능이 추가되고 있으며 인공 지능 기법을 이용한 자동 생산이나 자동 관리와 같은 지능적 기능의 지원이 시도되고 있다. CASE의 통합은 오래 전부터 시도 되어온 개념이지만 아직 개선의 여지는 남아있다. 도구들 간의 통합은 물론 라이프 싸이클(life cycle)측면의 통합도 이루어져야 하며 이것은 사용자에게 투명해야 한다. 완벽한 투명성이 보장되기 위해선 방법론의 조정이 필요할 것이다.

정보 저장소는 통합되어가는 추세이며 분산 처리와 다중 사용자 기능이 시도되고 있다. 본격적인 다중 사용자를 지원하기 위해서는 고전적인 세그먼크 단위의 체크인(check in) 체크아웃(check-out) 기능은 부족하며 DBMS 수준의 락킹(locking) 기법이 필요할 것이다.

앞으로 소프트웨어 공학인은 다양한 응용 분야에 대해서 잘 알고 있어야 하고 많은 방법론과 각각의 장단점 및 함정에도 익숙해야 하고 여러 CASE 도구, 프로젝트 라이프 사이클, 프로젝트 관리 기법, 하드웨어, 오퍼레이팅 시스템, 데이터베이스, 네트웍, 정보 저장소 등등에 대해서도 잘 알고 있어야 할 것이다. 소프트웨어 공학인은 만물박사가 되어야 하는가? 로마인처럼 되어야 하는가?