□ 기획연재 □

컴퓨터 과학 산책(7)

소프트웨어 개발과 신뢰성

한국전자통신연구소 고재상

소프트웨어 도전

정보화 사회에서는 정보를 대량으로 처리하는 정보시스템의 수요가 증가하게 된다. 정보시스템의 수요 증가는 컴퓨터 하드웨어 부분의 급진적인 발전 요인이 되었으며 지난 20년동안에 하드웨어 부분의 개발비용은 감소되었지만 정보를 처리하는 성능은 200배 이상 향상되었다. 고성능의 하드웨어 시스템을 개발할 수 있게 됨으로써 대형의 복잡한 소프트웨어 시스템이 요구되지만 소프트웨어 기술이 이를 따르지 못하여 소프트웨어 개발 및 보전에 막대한 비용이 소요되고 있는 실정이다. 또한 사내통신 수단이었던 근거리 통신망을 대체하는 인트라넷트가 더욱 확산되면서 소프트웨어 산업은 기업시장을 겨냥해 크게 성장할 것으로 전망된다.

세계 하드웨어 시장규모는 90년도 이후 총액규모에서 약간 감소하거나 거의 정체되는 현상을 나타내고 있는 반면에 소프트웨어 산업 성장 추이는 매년 15%의 성장율을 보이고 있어 2000년대 초에는 세계 시장규모가 9,600억 달러에 달할 것으로 전망하고 있다. 90년대에 들어와 소프트웨어가 하드웨어 대비 그 비중이 커지고 있는 것도 주목할 만한 사실이다.

국내에서도 1987년 이후 정보화가 꾸준히 진행되어 정보통신서비스 시장이 급격히 확대되면서 정보통신 소프트웨어의 수요가 크게 증가해 내수시장이 확대되고 있다. 1980년대의 디지털화와 규제 완화의 큰 흐름이 90년대 들어 멀티미디어화와 글로벌화로 바뀌면서 인터네트(INTERNET)서비스와 초고속정보통신서비스를 위한 정보통신 산업의 변화 속도는 급변하고 있다.

소프트웨어 품질보증

현재의 통신망은 소프트웨어 또는 하드웨어와 소프트웨어 복합된 프로그램을 가진 컴퓨터의 거대한 상호연결로서 묘사되고 있다. 소프트웨어 개발의 경이적인 성장때문에 "as much an art as a science"로 기술되고 있는 학문분야에 대해 품질관리 및 품질보증의 원리를 적용하기 위한 많은 연구가 계속되고 있다.

소프트웨어 개발의 중요문제 중의 하나는 시스템의 유효성을 평가할 수 있는 믿을만한 척도가 부족하다. 증명방법은 실행해 보는 것인데 이때 수행하고자 한 기능을 시스템이 수행하거나 또는 수행하지 못할 것이다. 새로운 시스템에서 결함이 발생하는 것은 피할 수 없기 때문에 모든 개발그룹은 결함을 발견하고 제거하는데 많은 시간을 소비하는 잠일에 당면한다.

이러한 문제를 볼 때, 하드웨어 설계 및 개발시 취해지는 동일한 품질보증 철학을 적용해야 한다. 즉, 결함을 발견하기 위해 소비되는 시간과 돈을 프로젝트 말기에 쓰는 것보다는 결함을 사전에 없애기 위해 프로젝트 초기에 투자된다면 보다 큰 성과를 얻을 수 있다는 것이다.

품질보증체계는 프로젝트의 성격 및 프로그래머의 창의력에 따라 개발노력을 감소시킬 수 있을 뿐만 아니라 프로젝트 개발때 자주 발생하는 변경사항을 관리할 수 있게 한다. 각 프로젝트마다 공식적으로 문서화하고 그 개발노력을 제품의 품질뿐만 아니고 경영노력의 품질도 보증할 수 있도록 분명하게 기술된 Checkpoints를 이용하여 엄격히 심사해야 한다.

소프트웨어 프로젝트에 있어서 품질을 개선

시키기 위해 다음과 같은 세 가지 품질보증 노력이 중점적으로 추진되어야 한다.

첫째, 소프트웨어 품질경영의 기본요소를 프로젝트 기발자들에게 인식시키고 실현하도록 지원하며, 품질은 그들의 책임이지 타조직에 위임될 수 있는 것이 아님을 주지시킨다.

둘째, 기존 개발방법의 타당성 및 적용성을 평가하기 위해 중요 소프트웨어 프로젝트에 대한 품질감사를 실시한다.

세째, 표준화된 품질척도(Quality metrics)를 측정방법과 표준과 함께 개발한다. 프로젝트의 소프트웨어품질을 정량화하는데 이용될 수 있으며 시간경과에 따른 개선사항을 추적하는데 활용한다.

결국, 품질보증계획서에는 개발 노력을 계량화시키고 측정할 수 있는 검사, 설계심사 및 품질감사(Audit)항목이 구체적으로 기술되어야 한다.

소프트웨어 품질문제에 대한 검토

소프트웨어 시스템의 품질 특성은 정확성(Correctness), 신뢰성(Reliability), 유지보수성(Maintainability), 유연성(Flexibility), 사용성(Usability), 효율성(Efficiency), 보안성(Security), 상호작용성(Interdoperability) 등을 말할 수 있는데. 소프트웨어 신뢰성 특성과 관련된 소프트웨어 품질설계기준은 Consistancy, Complexity, Accuracy, Error Tolerance 등이라 할 수 있다. 즉, 고신뢰성 프로그램은 정확하고, 완벽하고, 일관성있는 설계 특성이 요구된다.

소프트웨어 품질 문제를 검토하기 위한 방법 세 가지를 소개하고자 한다.

첫째, 검사(Inspection)는 소프트웨어 프로젝트의 문서 또는 코드의 정형적이고 구조적인 사항 및 문서화된 사항을 조사하는 것이다.

그 목적이 에러를 발견하는 것이기 때문에 에러를 검출하기 쉽고 그것을 교정하는데 소요되는 비용을 적게 하도록 검사는 프로젝트의 초기단계에서 실시하는 것이 보다 유익하다.

둘째, 소프트웨어품질을 검토하는 두번째 방법은 설계심사(Design review)이다. 검사와 마찬가지로 프로그램의 요구조건분석(Requirements), 구조분석, 설계 및 코팅하는 과정에서 프로젝트의 내적인 검토를 말한다.

설계심사가 자체적으로 규정된 조직 및 심사항목에 포함된 개별적인 사항의 전문성에 따라 보다 달라지는 반면에 검사(Inspection)활동은 세밀하게 작성된 조사항목을 가지고 공식적인 문서화를 추적하는 측면에서 차이가 있다.

세째, 소프트웨어 품질을 검토하는 세번째 방법은 "Walk-Through"이다. 말그대로, 프로그래머는 코드가 의도한 바를 성취할 거라는 것을 확인하기 위해 라인을 따라 소프트웨어 소스코드를 Walk-Through한다. 이 방법은 개발 중 어떤 시점에서라도 프로그래머가 스스로 필요하다고 느낄 때 사용될 수 있다.

검사, 설계심사 및 Walk-Through가 소프트웨어 품질을 개선시키는데 매우 유용한 것이기 때문에, Trouble을 피하기 위한 방법으로도 이용되고 있다. 일반적으로 이러한 Audit는 제품자체보다는 오히려 프로젝트의 경영에 중점을 두고 있다.

소프트웨어 신뢰성 개념

이와 같이 많은 비용이 소프트웨어 부분에 투자되면서 설계지원도구, 개발 및 관리체계, 형상 및 변경관리, 재이용 도구, 요구분석지원도구 등 설계 자동화 도구를 개발하여 사용하므로써 소프트웨어 생산성을 향상시키고 있다. 개발비용의 약 50%가 소프트웨어 시험비용에 소요되고 있지만 비용, 개발기간 등 사원의 제약으로 충분한 시험을 할 수 없을 뿐만 아니라 시험이 충분히 이루어졌다는 기준을 정하기 어려우므로 개발완료된 소프트웨어 품질을 보증할 수가 없고, 양도시간(Release time)을 결정하기 어렵다. 이를 계량적으로 결정하기 위해 소프트웨어 신뢰성 평가가 필요하다. 소프트웨어 신뢰도를 하드웨어적인 신뢰도의 4가지 요소(기능, 운용환경, 주어진 기간, 확률)와 같은 개념으로 정의하면 소프트웨어 시스템이 주어진 규격(요구조건)하에서 의도하는 사용기간 중에 고장 발생없이 만족스럽게 동작할 확률이라 할 수 있다. 이 정의는 에러나 결함의 형태에 따라 시스템에 미치는 치명적인 기능장해의 정도와 고장발생빈도를 고려하지 않은

단점이 있다. 이러한 단점을 보완하여 고장발생시의 손실((Penalty) 비용을 가중 인자로 소프트웨어 신뢰성을 정의하기도 한다. 또 시간 개념대신 실행횟수(Run)의 개념을 도입하여 주어진 규격하에서 실행할 때 고장이 일어나지 않을 확률로 정의하기도 하였다. 이와 같은 정의에서 보면 소프트웨어의 고장은 규격과 실제 결과가 다를 때를 의미한다고 볼 수 있다.

소프트웨어 신뢰도 모형

소프트웨어 시험방법은 프로그램의 모든 실행경로(Execution path)를 시험하는 Path oriented test와 각 입력들에 의한 Symbolic기능으로 출력을 얻는 Symbolic test, 그리고 입력공간에 대한 에러의 통계적 분포에 관한 가정을 기초로 시험하는 Probabilistic test로 분류된다.

이와 같은 시험에 의해 소프트웨어 신뢰성[신뢰도, MTTF(Mean Time To Failure), 잔존 에러수]을 추정하게 된다.

소프트웨어 신뢰성을 추정하기 위한 신뢰도 모형은 1970년대 초부터 Jelinski와 Moranda, Musa 등에 의해 발표되기 시작했으며, 소프트웨어 시스템을 Black-box로 간주하여 시스템 내의 모든 에러가 시스템 고장발생에 균등하게 영향을 미치고,시스템 고장 발생율은 시스템 내에 있는 에러수에 비례한다고 보았다.

신뢰도 모형은 접근방법에 따라 고장시간 모형(Time between failures model), 에러계수 모형(에러 count model), 에러파종 모형(에러 seeding model), 그리고 입력영역 모형(Input domain model)으로 분류할 수 있다.

고장시간 모형으로서 Zelinski/Moranda 모형, Littlewood/Berrall 모형, Schick/Wolverton 모형 등은 에러를 수정하면 확실히 수정된다고 가정했다. 결국 이 가정은 추론은 간편하나 항상 사실이 아니며, 가치없는 추정치를 얻게 된다. 이러한 한계를 극복하기 위해 Goel/Okumoto와 Kubat는 불안전한 수정 모형(Imperfact debugging model)을 제안하고 있다.