소프트웨어 開發 擴大 方案

이 기 호
梨花女大 電算學科 敎授

전략산업인
소프트웨어 산업은
사용자 및 경영자의 올바른
인식과 고급 인력에 대한 환경조성
시장의 확대와 다양한 기종의 개발 및
소프트웨어의 보호를 위한 저작권
그리고 소프트웨어의 관리·
보수·유지를 통하여
고부가가치산업으로
유도해야 한다.

소프트웨어라는 것이 하나의 독립된 단어가 되어 우리 귀에 익숙해지고, 소프트웨어 산업으로 인식되기까지 몇 년의 세월이 필요했다. 두뇌집약형 무공해 산업이니 부가가치가 높은 자원 절약형 산업이라느니 하는 말들을 서슴없이 하게 되었다. 앞으로 소프트웨어 개발은 무진장하고,세계 시장을 향해 뻗어나갈 가장 좋은 산업이라는 말을 많이 하고 있다.

그러기에 소프트웨어 개발이라는 과제에 많은 관심을 갖게 되고, 개발 촉진을 위한 방법을 모색케 되었다. 소프트웨어 개발은 거창한 구호나 법제도와 같은 외적인 조건의 추진만으로는 그 목적을 달성하기 어려운 것이다. 소위 고급 인력이라고 하는 자원이 바탕이 되어야 하기 때문이다. 그러기에 기업가의 센스나 열의만으로는 해결하기 어려운 것이다. 무한히 돈을 벌 수 있을 것 같으면서도 빨리 무엇인가 잡히지 않는데는 분명히 중요한 이유가 있을 것이다.

이제 소프트웨어의 산업으로서의 위치, 개발의 필요성, 중요성은 너무나 많이 논의가 되었기에 여기서는 더 이상 거론치 않고, 소프트웨어 개발이라는 대전제에 따른 문제점이라고 하는 것들을 검토해 보고 개선함으로써 문제 해결점을 찾아보고자 한다. 이것이 바로 소프트웨어 개발 확대 방안의 기조가 된다고 믿기 때문이다.

소프트웨어 개발이 잘 안되는 요인으로 고급 두뇌의 부족 현상, 좁은 시장 등을 들고 있다. 그러나 좀더 구체적으로 그 요인들을 찾아 내고 소프트웨어 개발에 따른 문제점들을 분석해 보고자 한다.

1. 소프트웨어에 대한 바른 인식

하드웨어에 반해 소프트웨어의 정의를 내린다면 간단히 말해서 프로그램이라고 할 수 있다. 이렇게 단순한 정의를 내리면 우리는 프로그램에 대한 잘못된 선입관이 있어 너무 가볍게 생각하는 경향이 있다.

무진장한 세계시장을 향한 전략산업으로 소프트웨어를 개발·육성해야 한다.

그러나 프로그램도 프로그램 나름이라는 것을 알아야 한다. 또한 프로그램을 작성하는 프로그래머도 최하위 직원으로 알고, 기술직이라고만 알고 있다. 프로그래머에도 계층이 있어 그 직위에 따라 엄연한 구별을 요한다. 일반이 생각하는 프로그래머는 단순히 프로그래밍 언어로 옮기다시피하는 코더와 혼동하는 경우가 가끔 있다. 하드웨어와 소프트웨어를 자동차와 기름으로 비유하는 경우가 있는데 이것은 완전한 이해는 아니다. 컴퓨터에 대한 기본지식이 없이는 소프트웨어를 완전히 이해했다고 할 수 없다.

따라서 컴퓨터를 최대로 이용하는 다양한 소프트웨어에 대한 이해도 완전하지는 못하다. 소프트웨어 개발 결과를 그 자체만으로 판단하는 것은 소프트웨어를 잘못 인식하는 것이다. 결과 형태가 몇장의 출력 용지일 수도 있고 디스켓 한 장일 수도 있고 자기테이프에 담아져 있는 몇피트의 테이프에 나타날 수도 있다. 잘된 프로그램일수록 사용이 편리하고 간단한 경우가 많은데 그럴수록 프로그램을 경시하고 컴퓨터를 요술장이로 만드는 경우가 있다. 프로그램을 작성한 프로그래머보다도 터미널에 앉아 그 프로그램을 이용하는 자가 더 위대해 보이는 것은 소프트웨어를 잘못 인식하는 증거이다.

예를 들어 중소 소프트웨어 하우스들이 일감 창출에 난색을 보이는 듯 하는 것도 소프트웨어를 완전히 이해 못한 증거이다. 중소 기업들이 정부 출연기관들에 일감을 빼앗긴다고 생각하는 이유가 여기에 있다고 본다. 소프트웨어는 두뇌 움직임의 결과이기에 일감을 찾는 것도 두뇌 작용의 일부로 생각된다.

2. 고급인력을 위한 환경 조성

고급인력이란 아마도 대학수준 이상의 교육을 받은 자를 지칭하는 경우가 허다할 것이다. 대학교육 수준이라면 전공을 가진 사람을 생각하게 된다. 소프트웨어 개발에 있어 필요한 고급 두뇌라고 하면 전산분야 전공자를 지칭할 수도 있다. 소프트웨어가 무엇인가를 확실히 안다면 누가 어떻게 해야 하는가를 알게 될 것이다. 소프트웨어 개발은 두뇌의 움직임으로 많이 이루어지기 때문에 주위 환경을 개선해 주는 것이 기본적인 조건이라고 할 수 있다.

소프트웨어 개발을 위한 전문인을 채용했으면 그의 전공에 맞게 일을 주는 것이 효율성을 높이는 것이다. 기업체가 전문인을 채용해서 영업이나 시키고 사무실을 여러 사람과 공동으로 사용하도록 하는 것은 비능률적이다. 소프트웨어 개발은 불연속적인 시간에 의해서는 이루어지기 어렵다. 몇날 몇일 밤을 하나의 소프트웨어 설계를 위해 고민하고 생각해야 하는 때가 있다.

그러기에 소프트웨어설계는 하나의 창작물이라고 한다. 과학과 예술의 조합이라고 하는 것이

바로 이러한 소프트웨어 설계 과정을 두고 말한다. 하나의 작품을 위해서는 골똘히 생각할 시간이 필요하며 조용한 장소가 필요하다.

그래서 외국에서는 이미 오래전부터 소프트웨어 개발자에 대해 이해를 하고 특별 대우를 해주었다. 단순 노동자가 아니라는 것을 깊이 인식하게 되었다. 사무실이 다른 직원과 공동 사용이 아니고 혼자 조용히 쓸 수 있게 되어 있고, 그 속에서 단절없이 사고할 수 있는 기회를 주는 것이다.

3. 소프트웨어의 상품화

소프트웨어는 개발하는 데만 목적이 있는 것이 아니라, 그 결과를 실체화하여 여러 사람이 여러 면에서 응용, 적용하는 데 의미가 있다. 따라서 개발 소프트웨어는 상품화되어야 하는데 그렇게 하기 위해서는 완전한 프로그램 패키지를 만들어야 한다. 즉, 어느 특정한 경우만을 위한다든가 제한점을 많이 둔다면, 일반성이 부족하여 상품으로서의 가치가 없을 것이다. 어떤 경우에도 오류 발생 가능성이 없어야 완전한 소프트웨어 패키지로서의 가치가 있는 것이다.

그래서 하나의 프로그램이 상품으로 등장하게 된다. 미국에서도 3~4년 전부터 프로그램만 파는 상점이 생겼다. 마치 레코드판을 파는 상점을 연상하면 될 것이다. 프로그램 선전도 가수가 레코드판을 낼 때 그 가수의 얼굴을 내놓고 선전하듯이 프로그램 디스켓에도 프로그래머의 얼굴을 넣어서 소개도 하며 다양한 선전술을 쓰고 있다.

4. 소프트웨어 시장과 다양한 기종 문제

소프트웨어를 개발, 상품화하여 내놓으면 팔 수 있는 시장이 있어야 함은 당연한 일이다. 외국을 상대로 수출하는 경우와 국내 수요를 생각할 수 있는데 모두 소프트웨어의 크기나 대상의 차이는 있겠지만 결과적으로는 같은 문제가 야기될 것이다. 여기서는 국내 시장만을 생각해 본다. 누구를 상대로 개발할 것인가, 무엇을 개발할 것인가를 생각해야 하는 데 가장 큰 문제는 어떤 기종에 맞출 것인가가 문제이다. 특정 기종만을 상대로 개발하기에는 그나마 작은 시장이 더 좁아진다.

현재 보급되어 있는 개인용 컴퓨터들은 디스크 드라이브가 아닌 카세트가 더 많은 실정이다.

카세트용 소프트웨어 개발을 하기에는 시대에 너무 뒤떨어진 것이다. 속도가 너무 느리고 오류 발생률이 많아 외국에서는 거의 사용치 않고 있기 때문이다. 다양한 기종간의 표준화된 것이 있고 기본적으로 디스크 드라이브가 부착된 것이라면 이러한 개인용 컴퓨터를 대상으로 한 소규모의 소프트웨어 개발이 활발해지고 차츰 정규 상품으로 등장하게 되면 소프트웨어 개발에 활력소가 될 것이다.

하드웨어와 달리 소프트웨어의 개발 중복성을 초반부터 걱정할 필요는 없다. 레코드판을 수십장 갖고 있듯이 프로그램이 같은 종류의 것이라도 프로그래머가 달라지면 또 살 수도 있기 때문이다.

5. 전산전공과 타전공간의 연계성

개발해야 할 소프트웨어는 무진장이라고 하면서 실제로 구체적인 대목은 이야기하기 어려운 점이 많이 있다. 이유인즉 전산인은 컴퓨터에 관한 것밖에 모르고 경영인은 경영에 관한 것, 교육자는 교육에 관한 것, 토목공학자는 토목에 관한 것밖에 모르기 때문인 경우가 많다. 어느 분야에서건 컴퓨터를 이용해서 문제 해결을 하는 데 효율성을 높일 수 있는 것은 사실이다.

그런데 예를 들어 건축분야라고 하면 그 분야 전공자만이 그들이 필요한 것을 정확히 알 수 있는데 전산학 지식이 없기 때문에 무엇을 어떻게 개발해야 할지 핵심을 잡지 못하는 경우가 있다.

개발 의뢰도 전산지식이 있어야 전문적인 소프트웨어 개발 요구를 정확히 할 수가 있다.

따라서, 각 분야의 전공자가 전산교육을 받았다면 그 사람이 그 분야를 개발하는 데 참여해서 전산전공인과 협조하는 것이 가장 좋은 경우가 되겠다. 그렇게 되면 전산전공자는 본래의 기능으로 시스템 소프트웨어를 개발 유지하는 데 종사할 것이고 고도의 기술을 요하는 부분에 대해서만 소프트웨어 개발을 도울 것이다.

6. 소프트웨어의 저작권 문제

컴퓨터를 구입하는 사람이 프로그램을 그냥 달라고 할 때 답답하다고 한다.

물론 메이커에서 주는 몇가지 기본적인 프로그램이외의 것을 요구하는 것이다. 소프트웨어에 대한 가치와 인식부족에서 오는 현상이다. 소프트웨어가 과학과 예술의 조화로 이루어진 저작권이라는 것이 확실하기에 저작권을 인정해서 개발자로 하여금 그에 해당하는 이익을 볼 수 있도록 만들어 주어야겠다. 함부로 복사해서는 안된다는 것은 상식이지만 고도의 기법으로 복사가 쉽게 되지 않도록 소프트웨어를 만드는 것이 더 중요하다. 메이커에서는 수십종의 프로그램을 개발했다고 선전하지만, 전문가의 입장에서 보면 저급의 하찮은 프로그램들이 너무나 많은 것같다.

이러한 소프트웨어를 놓고 법적 보호로부터 바라지 말고 만들어 놓은 소프트웨어를 많이 이용해서 그 기법이나 내용, 효율면에서 객관적인 태도가 필요하다고 본다. 그 결과로 좀더 세련되고 전문성있는 프로그램 개발의 기틀을 잡았으면 한다.

7. 대형 소프트웨어 개발

세계를 향한 소프트웨어 수출이나 대형 전문적 소프트웨어를 목표로 삼는 소프트웨어 전문업체라면 소프트웨어 프로젝트 수행을 위한 개발팀 구성원의 조직에 신경을 써야 할 것이다. 규모가 크고 전문성이 있는 소프트웨어 개발은 소규모 소프트웨어 개발의 양상과는 전혀 다른 문제점들이 발생하기 때문에 소프트웨어 개발자들의 구성 조직이 중요한 역할을 하는 것이다.

소프트웨어 개발자들의 능률이 곧 그 기업체의 능률이요, 이익과 관계되기 때문에 어떻게 능률적인 개발을 하도록 유도할 것인가가 고려되어야 할 문제이다. 이 부분이 근래 소프트웨어 공학의 일부분이 되어 있기도 하다. 대형 소프트웨어는 여러팀이 분업으로 해야 하기 때문에 팀구성과, 한 팀에 해당하는 구성 인원의 적정수 결정 등이 문제가 된다. 구성 인원이 너무 많으면 구성원간의 의사 소통을 위한 시간 낭비가 많고 소프트웨어 테스팅이 어렵고 프로그램 연결에 있어 오류 발생률이 높다.

소단위 개발팀은 팀의 수준이 향상되고 구성원들간에 밀접한 관계가 이루어져 서로 소통이 잘되고 서로 다른 구성원의 일을 이해해서 한 사람의 결원이 생긴다해도 계속 연결지을 수 있는 장점이 있다. 또 소단위 팀 구성은 구성원 대부분이 숙련되고 유능해서 일처리가 잘되는데 대개 유능한 사람은 주로 소프트웨어 개발에 참여치 않고 관리자 위치에 서게 되므로 사실상 팀 구성원들은 숙련되지 못한 사람들로 이루어지는 경향이 있어 재고해야 할 문제라고 생각된다.

8. 요구 사양의 명확성

소프트웨어 개발을 위해 처음 단계로 중요한 것이 무엇을 개발하고자 하느냐 하는 문제 분석이다. 이 문제 분석에 앞서 중요한 것이 정확한 문제 제기이다. 요구하는 것이 무엇인지 그 사양을 구체적이고도 명확히 밝히고 어떤 결과를 기대할 것까지도 요구하는 것이 문제 해결을 정확히 하게 한다.

또 전문인을 채용해서 그 전공에 맞게 일을 부여하는 것이 고급인력의 낭비를 막는 경우가 될 것이다. 고급인력의 부족이라고는 하지만 절대수를 따지지말고 우선 현재의 고급인력만이라도 충분히 활용할 줄 아는 관리자가 필요하다. 일반 사원 다루듯이 소프트웨어 개발자를 다루면 그 진행 속도는 대단히 느리고 비효율적이며 생산성 저하의 원인이 된다. 적어도 소프트웨어 개발팀의 팀장은 소프트웨어가 무엇인가를 확실히 알고, 무엇을 개발할 것인가 알려줄 뿐 아니라 진행중인 소프트웨어 개발에 가장 적절한 기법을 제시하고 검토할 수 있는 능력이 있어야 한다.

9. 소프트웨어 관리 · 보수 · 유지

소프트웨어의 수명(Life cycle)이 하드웨어의 수명에 비해 길다는 이유로 육성할 가치가 있는 미래 산업으로 여겨진다. 오랜 기간에 걸쳐 개발한 소프트웨어가 그 수명이 짧다면 투자에 대한 이윤 가치가 너무 없게 된다. 따라서 소프트

웨어 수명을 길게 하는 것이 또 하나의 과제이다.

그래서 소프트웨어 설계 공법 도입이 필요하고 프로그램의 올바른 기법 채택이 중요하다.

소프트웨어를 개발한 후 적절히 유지 계승하는 것이 전체적인 비용 절감을 가져온다.

근래는 소프트웨어 개발에 드는 비용과 더불어 유지 보수하는 데 드는 비용이 점차 증가하고 있다. 따라서 유지 보수하기 위해서 정확한 다큐멘테이션 작성이 필수적이며 적용에 불편이 없고 또 보수하는 데 어려움이 없도록 처음 설계와 구현할 때 시간과 노력을 더 투자하는 것이 비용 절감을 위해 더 현명하다고 하겠다.

10. 결 론

소프트웨어 개발은 결국 고급인력이라고 하는 소프트웨어 개발자 즉, 프로그래머가 할 것이다.

유능한 경영자나 관리자가 하는 것이 아니다. 그러므로 소프트웨어 개발자를 잘 다룰 줄 아는 관리자가 필요하다. 높은 차원에서의 적절한 업무의 배분, 소프트웨어 공법, 프로그램 기법을 알고 적용할 줄 아는 능력이 있는 관리자라야 한다.

경영자는 소프트웨어 개발자에 대한 인식을새롭게 하고, 업무의 성격상 특수성을 인정하여 환경을 개선해 주고 성급한 재촉보다는 일정기간 투자를 계속해야 할 것이다.