이 논문은 소프트웨어 분야가 어째서 많은 개발 비용을 필요로하며 발전 속도가 느린 이유를 말하고, 이러한 문제를 해결하는 방안을 모색하고 있다.

저자는 그동안 소프트웨어 업계에 많은 변화가 있었고 많은 기술일 도입됐지만 이들이 소프트웨어 분야가 갖는 본질적인 문제를 해결하지 못했다고 생각한다. 소프트웨어 기술이 도입된 이래로 고급언어, 시분할 시스템, 객체 지향, 인공지능, 자동 프로그래밍 등의 시스템이 등장했다. 저자에 따르면 이는 모두 좋은 기술인 것은 맞지만 소프트웨어가 갖는 부수적인 문제점만을 해결해주고 본질적인 문제는 해결해주지 못한다. 즉, 개발자가 저급 언어에서 벗어나 고급언어를 사용하는 것과 절차지향 언어에서 벗어나 객체지향 언어를 사용하는 것은 작은 문제를 해결한 것이고, 시분할 시스템과 인공지능, 자동 프로그래밍은 보편적이라기 보다는 특정 분야에만 특화되어 있다는 것이다.

동시에 저자는 소프트웨어와 하드웨어의 발전 속도를 말한다. 저자에 따르면 하드웨어의 발전 속도가 이례적으로 빠르다. 또한 소프트웨어는 눈에 보이지도 않고 만질 수도 없어서 태생적으로 복잡하다. 실체가 없기에 소프트웨어는 개발자로 하여금 더 많은 노력과 능력을 요구한다는 것이다. 그리고 소프트웨어는 그 유용성이 확인되면서 다양한 분야에서 다양한 요구를 받고 있다. 안그래도 개발이 힘든데 소프트웨어의 적용 분야가 다양해짐에 따라 보다 다채롭게 개발해야 하는 것 역시 소프트웨어의 발전 속도를 늦춘다.

사람들이 소프트웨어에 대한 이해도가 떨어지는 것 역시 한 몫 한다. 정확히는 고객들이 자신의 요구사항을 정확히 모르고 의뢰를 하며, 개발자 역시 고객의 정확한 주문을 모르고 개발하는 것이 문제다. 따라서 소프트웨어 발전은 느리고 불완전할 수 밖에 없다.

이에 따라 저자는 소프트웨어 발전의 본질적인 문제를 해결하기 위한 방안을 제시한다.

우선 소프트웨어를 전부 개발하려고 하지 말고 다른 소프트웨어를 구매해야 한다. 현재 시장에는 다양한 판매 업체가 있다. 그들에게 소프트웨어를 구매하면서 동시에 소프트웨어의 유지 및 보수도 그들의 몫으로 두어야 한다. 다른 업체의 소프트웨어를 구매하고 사용함으로써 개발 비용과 시간을 절약하는 것이다. 타 산업에서 한 기업이 원자재의 생산, 정제 및 조립을 모두 하는 것이 아니라 여러 업체로 나눠서 협업하는 것처럼 말이다.

그리고 개발자는 고객과 꾸준히 소통해야 한다. 고객은 서비스의 구매자로서 왔지만 그들은 자신의 교구사항을 자세히 모른다. 따라서 개발자는 고객과 꾸준히 소통하면서 그들의 요구사항을 반복적으로 확인해야 한다. 반복적인 확인을 통해 고객의 목적과 실현 방법을 구체화하고 프로그램을 개발함으로써 프로그램 수정을 줄이는 것이 중요하다.

또한 점진적 개발 역시 중요하다. 일단 프로그램은 단순하더라도 형태가 있어야 한다. 구현된 프

로그램을 통해 개발자는 영감을 받을 수 있다. 그 뿐만 아니라 개발자들은 중간 결과물을 지속적으로 확인하면서 의욕이 떨어지는 것을 방지할 수 있다. 처음부터 완벽한 추상화는 존재할 수 없고 따라서 처음부터 완전한 개발은 불가능하다는 것을 명심해야 한다.

마지막으로 소프트웨어 업계는 좋은 설계자를 육성해야 한다. 좋은 설계자란 때론 경험과 무관할 수도 있다. 따라서 기업은 개발자들에게 멘토를 붙임으로써 개발자가 수월하게 학습하고, 동시에 기업은 좋은 설계자를 찾을 수 있도록 해야 한다. 또한 개발자들끼리 상호작용하면서 동료로부터 배우고 자극을 받을 필요가 있다.

이 논문을 읽고 난 소프트웨어에 대한 인식 개선이 필요하다는 생각이 떠올랐다. 우리는 소프트웨어가 추상적이고 눈에 안 보인다고 가볍게 여기면 안 된다. 소프트웨어 역시 건축과 자동차처럼 엄연히 산업의 한 분야이고 개발과 사용에 많은 노력이 필요하다는 인식이 필요하다. 그 동안 소프트웨어는 비가시성과 수정이 용이하다는 점으로 인해 사용자들로부터 가벼운 인식을 받았던 것 같다.

따라서 소프트웨어를 너무 세분화하고 커스터마이징할 필요가 없다고 생각한다. 자동차와 냉장고 같은 다른 제품처럼 사용자가 프로그램에 적응하는 것이 필요하다. 물론 개인 맞춤형 프로그램은 소프트웨어의 장점 중 하나이다. 하지만 소프트웨어 업계의 발전과 체계화를 위해서 우선 사용자들의 인식 개선과 보편화된 기준점으로서의 프로그램이 필요하다고 생각한다. 이는 사용자들의 편의성으로 곧바로 이어질 것이다.

또한 산업의 다양화가 필요하다. 한 기업에서, 팀에서, 프로젝트에서 프로그램을 세분화하기 보다는 현재 다른 공업처럼 업체를 나눌 필요가 있어 보인다. 하청 업체와 유지 및 보수 업체를 따로 두고 개발 업계는 그들의 분야에만 집중해야 한다. 그리고 저자가 말한 것처럼 다른 기업에서 만든 프로그램을 구매하고 사용하는데 있어서 주저함이 있으면 안 된다.

이를 위해서는 개발자 뿐만 아니라 일반인들에게도 소프트웨어 교육이 필요하다. 현재 산업 흐름은 소프트웨어 쪽으로 넘어왔다. 대부분의 사람들이 소프트웨어의 잠재적인 고객이 될 것은 자명하다. 따라서 교육업계에서 간단한 언어와 개발을 교육시켜야 한다. 10년 전, 중국이 세계 시장의 중심 국가로 떠오르자 많은 학교에서 교양 강의로 중국어를 가르치는 것처럼 말이다. 현재 숭실대학교에서 1학년 학생을 대상으로 소프트웨어 교육을 실시하고 있는 것은 바람직하다고 생각한다.

인류는 세상을 추상화하기 위해서 수학과 철학, 물리를 사용하였다. 그러나 우리는 인류의 역사가 수 천년을 넘었음에도 불구하고 세상을 밝히지 못했고 오히려 나아갈 길이 훨씬 많다는 것만 깨달았다. 소프트웨어는 인간의 생각을 현실로 구현하는 것이다. 따라서 세상을 완벽하게 추상화하지 못한 인류가 자신의 생각을 현실로 구현하는 것이 힘든 것은 당연하다. 따라서 우리는 인류

가 불완전한 것처럼 우리의 소프트웨어가 불완전하다는 것을 이해하고, 소프트웨어에 발전을 참을성 있게 기다리며 우리 역시 소프트웨어를 이해하기 위해 노력할 필요가 있다.