Chapter 05

제품 관리 vs. 엔지니어링

| 적절한 제품 빌드하기 vs. 적절하게 제품 빌드하기 |

실제 고객의 니즈, 이제 막 가능해진 솔루션, 이 둘이 합쳐진 결과가 뛰어난 제품이라면 제품 관리자와 엔지니어링 팀의 관계가 왜 그렇게 무척이나 중요한지 쉽게 이해할 수 있다.

제품 관리자는 솔루션 정의를 책임지지만 어떤 것이 가능한지 가장 잘 알고 있는 쪽은 엔지니어링 팀이므로, 엔지니어링 팀이 솔루션을 최종적으로 전달해야 한다. 제품 관리자로서 당신은 엔지니어링 팀과 좋은 관계를 유지해야 일 자체도 잘 해결될 수 있다는 사실을 모르지 않는다. 만일 관계가 불편하다면 험난하고 기나긴 시간들을 맞이할 것이다.

좋은 관계를 향한 비결은 서로를 동료라고 이해하는 것이다. 어떤 직책도 다른 직책에 종속되지 않는다. 제품 관리자로서 당신은 적절한 제품을 정의할 책임을 맡고, 당신의 엔지니어링 책임자는 적절하게 제품을 만들 책임을 맡는다. 이 두 가지는 모두 필요하다. 당신은 엔지니어링 책임자가 뛰어난 제품을 빌드하는 데 필요하다고 판단한 것을 직접 진행할 수 있도록 해줘야 하고, 엔지니어링 책임자는 '가치 있고 사용할 수 있는 제품'을 당신에게 제시해야 한다.

양쪽 다 서로에게 커다란 도움이 될 수 있다. 구체적으로 말하면, 엔지니어들은 이기는 제품을 정의하려고 노력하므로 당신에게 큰 도움이 될 수 있다. 엔지니어들은 무엇이 가능한지 어느 누구보다도 잘 알고 있다는 사실을 기억해야 한다.

다음은 엔지니어들이 더 나은 제품을 내놓을 수 있도록 하는 세 가지 방법이다.

1. 엔지니어들을 사용자와 고객 앞으로 다가가도록 하라. 사용자들이 고심하는 모습을 지켜보면 많은 것을 배울 수 있을 뿐만 아니라 문제를 더 진중하게 제대로 이해할 수 있다. 그래야 훨씬 더 나은 아이디어와 솔루션을 찾기 위한 자극도 받을 수 있다. 프로토타입 테스트에 엔지니어를 합류시키는 것부터 가볍게 시작하라.

2. 기술의 발전에 따라 무엇이 가능해지는지를 파악할 때 엔지니어들에게 도움을 구하라. 이미 실현 가능한 기술과 가능해질 기술들을 서로 구별하여 이들 기술이 문제를 해결하는 데 어떤 도움이 될지 브레인스토밍하라.

3. 제품 발굴 프로세스가 시작할 때부터 엔지니어들(적어도 선임 엔지니어 한 명)을 참여시켜 아이디어의 상대적 비용을 평가하고 더 나은 솔루션을 찾아내는 데 활용하라. 제품 관리자는 뛰어난 제품을 정의해 놓고 사전 협의도 없이 엔지니어링 팀으로 넘겨버리는 실수를 저지르곤 한다. 그러면 필요한 것과 가능한 것을 협의하는 매우 중요한 과정이 늦춰지고 결

국에는 제대로 된 결정을 내릴 시간이 남지 않게 된다.

마찬가지로 당신도 엔지니어들에게 커다란 도움이 될 수 있다. 다음은 엔지니어들에게 도움이 될 수 있는 방법 세 가지다.

1. '최소한'의 제품에 집중하라. 이 점에 관해서는 뒤에서 자세히 언급하겠지만, 제품 관리자로서 당신의 일은 최종 제품을 정의하는 것이 아니라 목표를 만족하는 최소한의 제품을 정의하는 것이다. 이 점 하나만으로도 제품 관리와 제품 엔지니어링 사이의 역동성을 근본적으로 개선할 수 있다.

2. 엔지니어링 팀이 제품 개발에 착수하면 바로 그 순간부터 변동(churn)을 최소화하기 위해 최선을 다하라. 변동은 제품의 요구사항이나 정의를 바꾸게 한다. 물론 불가피한 변동도 있고, 엔지니어들도 자신들의 통제를 넘어서는 것이 있다는 점을 이해하고 있다. 하지만 더할 나위 없이 대단한 최신 아이디어라며 시험해보기에는 이미 늦었다.

3. 구현 단계에서는 이런저런 의문들이 필연적으로 발생한다. 개중에는 놓쳤거나 완벽하게 생각을 마무리 짓지 못했던 유스 케이스도 포함된다. 이는 정상적인 과정이며, 최고의 제품 팀에서도 마찬가지다. 하지만 구현 단계가 진행되는 동안 제품 관리자로서 당신이 해야 할 임무는 문제에 직접 뛰어 들어 가능한 한 신속하게 해답을 얻어내고 최소한의 제품에 늘 집중하며 변동을 최소화하는 것이다.

이 점을 염두에 두고 나는 늘 최고의 엔지니어들을 독려하여 그들이 제품

관리에도 관여할 수 있도록 애쓰고 있다. 그리고 빌드할 만한 가치가 있는 것을 엔지니어링 팀이 받지 못한다면 엔지니어링이 얼마나 위대한지는 아무짝에도 쓸모가 없다고 주장한다. 더불어, 무엇을 빌드하느냐와 같은 큰 문제를 직접 해결하며 무엇이 가능한지를 알게 된 엔지니어에 의해 만들어진 좋은 제품이나 회사들을 일일이 거론할 때도 있다. 이는 엔지니어 자신의 경력 개발에도, 제품 자체에도 대단히 중요한 것이다(또한 고객과 회사에도 마찬가지다).

? 원격 개발자와 함께 성공할 수 있는 방법은 무엇일까?

오늘날 가장 흔히 접할 수 있는 상황 가운데에는 제품 관리자와 엔지니어링 팀의 일터가 서로 다르다는 것이 있다. 인도 등지에 아웃소싱하는 것만을 이야기하는 것이 아니다. 원격 개발 팀은 인수 합병으로 생겨나기도 하고, 조직 자체의 규모가 커서 개발자들이 다른 건물에서 일하기도 한다.

개발자들이 바로 옆 자리에 앉아 있지 않는다면 일반적인 의사소통이나 일 처리에서 부딪히는 사소한 문제가 커지기도 하고, 경우에 따라서는 원격 개발이라면 고개부터 설레설레 흔드는 상황이 심심치 않게 벌어지기도 한다. 또한 원격 개발이 가져다주는 원가 절감을 당연시 하지 않고 문제를 제기하는 팀원도 생겨날 수 있다.

원격 개발 팀과 일하는 경우, 성공 확률을 획기적으로 높일 수 있는 다음 세 가지 핵심 사항을 고려하는 것이 좋다.

1. 개발 팀이 멀리 있을수록, 다시 말해 언어적, 문화적, 시간적 차이로 인한 소통 문제가 많아질수록, 제품 사양에 기울이는 업무가 완벽해야 한다는 중압감은 더 커진다. 무엇을 빌드할지 제품 관리자 스스로가 확신하지 못하고(또는 갈피를 잡지 못해 계속 마음을 바꾸고), 원격 엔지니어링 팀이 우왕좌왕할 때 악몽 같은 프로젝트가 시작된다. 마치 칼자루가 아닌 칼날을 쥐고 맛없을 것이 뻔한 레시피로 요리를 하는 격이다.

   '제품 사양 다시 만들기' 장에서 나는 제품 사양의 기준으로서 완성품과 거의 동일한 프로토타입의 중요성을 언급할 것이다. 여기서는 그 장의 내용을 미리 반복

하지는 않겠지만, 이 말은 꼭 언급하고자 한다. 팀이 멀리 떨어진 곳에 있다면 완성품과 거의 동일한 프로토타입을 주요 의사소통 창구로, 즉 실제 사양뿐만 아니라 변경 내용까지 소통하기 위한 창구로 사용해야 한다. 문서(제품 사양)를 통해 소통한다면 문서에 현지 언어가 쓰일 수도 있고, 사람들에게 일일이 문서를 읽으라고 할 수도 없는 노릇이라, 작성자가 바로 아래층 사무실에 앉아 문제를 분명하게 설명해 주지 않는다면 골치만 더 아플 뿐이다.

2. 개발 팀과 함께 일한다면 자원이 충돌(가령, 관리자 두 명이 상반되는 지시를 내리는 경우)하더라도 곧바로 해결된다. 반면 원격 팀일 때는 달갑지 않은 놀라움을 자주 겪을 수 있고, 말 그대로 몇 달 동안 문제를 파악하지도 못한 채 시간만 지나갈 수도 있다. 이는 대개 원격 개발자들이 누가 무엇을 원하는지 그리고 서로 다른 지시를 어떻게 해석해야 하는지에 대해 가정할 수밖에 없기 때문에(또한 가정 자체도 올바른 경우가 드물기 때문에) 일어나는 일이다. 따라서 함께 있는 누군가가 원격 팀과의 모든 협력을 관리하게 하는 것이 대단히 중요하다. 그렇다고 해서 의사소통 전체가 이 사람을 통해야 한다는 것은 아니지만, 엔지니어링 팀이 누구의 책임 아래에 있어야 하는 것에는 이견이 없어야 한다. 이런 책임을 맡은 사람은 프로젝트 관리자(프로그램 관리자라고도 할 수 있다)일 수도 있고, 디렉터나 엔지니어 부책임자일 수도 있다. 이 경우 엔지니어링 부책임자는 제품 관리자와 멀리 떨어지지 않은 곳에 있어야 한다.

3. 오늘날 대부분의 사업체 내에서 사용할 수 있는 훌륭한 의사소통 메커니즘은 많이 존재한다. 이메일 및 인스턴트 메시지 소프트웨어 외에도 화상 회의 기술이 비약적으로 발전했고, VoIP 또한 국제 통화 비용이 대폭 낮아졌다. 그렇긴 하지만 개인이 서로 얼굴을 맞대고 소통하는 관계를 대신할 만한 것은 없다. 적어도 분기당 한 번쯤은 제품 관리자가 사무실을 박차고 나가 엔지니어링 팀과 얼굴을 맞대고 의미 있는 시간을 보내야 하며, 주요 아키텍트들과 관리자들도 만나야 한다. 이런 대면 방문으로 서로의 관계와 소통의 질은 향상된다. 원활한 소통을 위한 또 하나의 방법은 주요 개발자들이 제품 관리자와 함께 하는 교환 프로그램, 또는 반대로 제품 관리자가 개발자와 함께 하는 교환 프로그램을 마련하는 것이다. 재능 있는 팀과 함께 하고, 앞서 설명한 관계를 관리하게 되면 실제로 사람들과의 약속이 즐거워진다.

특히 엔지니어들이 인도에 거주한다면 서로 다른 시간대 때문에 아침에 출근하여 전날 진행된 결과를 받아들고 검토, 테스트하며 피드백을 보내느라 여념이 없는 일

상을 반복할 수 있다. 지극히 빠른 사이클 타임인 셈이다.

프로토타입 관련 자원도 원격지에 둘 수 있다는 점을 유의해야 한다. 하지만 이 경우는 더욱 빠른 사이클 타임이라서 의사소통에 더 노력해야 하고 업무 시간을 더 유연하게 활용해야 한다(하루에도 전체 업무가 몇 번씩 반복된다).

원격 개발 팀과 작업할 때 일어나는 문제를 해결할 수 있는 또 하나의 해결책으로 제품 팀 전체를 원격지에 함께 두는 방식도 고려해 볼 만하다. 나는 이제 막 등장한 이 방식이 점점 더 보편화될 것으로 본다. 하지만 아직은 고민하기 이르다. 원격 엔지니어링 및 QA가 자리 잡는 데에도 10년은 걸렸으며, 숙련된 제품 관리자와 디자이너까지 자리 잡으려면 10년은 더 걸릴 것이다.

? 아웃소싱은 어떤가?

내가 지금 언급하는 모든 회사는 아웃소싱이 어느 정도 자리를 잡았다. 그러나 그 결과는 제각각이다. 나는 이들 회사가 직면한 문제에 몇 가지 원인이 있다고 생각한다. 종종 이들 문제는 제품 개발 프로세스나 언어적, 문화적 요소에서 비롯되지만, 대개는 문제의 핵심이 아웃소싱을 잘못된 근거로 활용한 데에서 출발한다.

감동적인 제품을 만들고자 한다면 어떤 부문의 아웃소싱을 원가 절감이라는 목적으로 진행하면 안 된다. 아웃소싱은 제품에 적절한 인력을 모아주는 것이 목적이어야 한다. 최고의 인력을 가까운 곳에 두지 못할 때가 무척이나 많다. 다시 말해 필요한 인력을 다른 지방에서 또는 다른 나라에서 채용할 수 있어야 한다.

실리콘 밸리는 물가가 매우 높아서 채용하려는 사람들에게 이곳에서 삶을 영위할 수 있을 만큼 보수를 지급하기가 어렵다. 이는 실리콘 밸리의 슬픈 진실이다. 다른 지역에서 출퇴근해야 어느 정도 타산이 맞는다. 결국 적절한 팀을 다른 곳에서 찾아봐야 한다.

다행히 뛰어난 제품 관련 인력을 찾을 수 있는 멋진 곳이 몇 군데 있다. 가령, 인도, 동유럽(특히 체코 공화국, 헝가리, 폴란드, 슬로바키아 등), 북유럽(특히 네덜란드, 스웨덴, 독일 등), 이스라엘, 중국, 싱가포르, 호주, 뉴질랜드 등이다. 이들 지역마다 놀랄 만한 능력의 소유자들이 반드시 존재한다. 내 경우에도 최고의 실력을 보이며 함께 일하는 것만으로도 일종의 특권을 주었던 팀원들이 스웨덴, 실리콘 밸리, 보스턴,

인도 등지에 흩어져 있었다. 그들과 함께 2천만 명이 넘는 사용자를 지원해야 할 인프라 소프트웨어를 만들었고, 재능을 갖춘 그들이 없었다면 성공은 일찌감치 남의 말이 되었을 것이다.

개인적으로 좋아하는 회사들 가운데 MySQL이 있다. 오랫동안 그런 철학을 구체화해 오고 있는 이 회사는 실리콘 밸리와 스웨덴이 명목상 본거지이지만 제품 팀, 그리고 중역진까지도 전 세계 곳곳에 흩어져 있다. 진정한 가상 조직이라 할 수 있는 이들은 곳곳에 흩어져 있는 최고의 데이터베이스 및 시스템 소프트웨어 인력이 가져다주는 장점을 누리고 있다. 완전히 분산된 제품 팀을 관리하기란 쉬운 일이 아니지만, 그들 전체가 어느 한 곳에 있었다면 그리고 틀에 박힌 중앙집중형 회사가 되려고 했다면 지금과 같은 결과를 달성하지 못했을 가능성이 매우 높다.

80년대에 제조업이 실리콘 밸리에서 밀려났던 것처럼, 오늘날에도 몇 가지 다른 직업군이 실리콘 밸리를 떠나고 있다. 특히 고객 서비스와 QA, 그리고 정도는 덜하지만 엔지니어링이 바로 그 주인공이다. 요즘에는 아키텍트와 QA 관리자를 제품 관리 및 디자인 관련 인력과 함께 본사에 배치하고, 나머지 인력을 세계 곳곳에 함께 또는 따로 배치하는 추세가 보편화되고 있다.

모든 것은 팀과 팀을 구성하는 개개인의 능력에 달려있다는 사실을 인식해야 한다. 많은 관리자가 이 점을 이해하지 못하고 있고, 다른 사람들보다 20배나 더 나은 생산성을 이룰 수 있다는 사실을 알고는 정신이 멍해지기도 한다. 뛰어난 능력으로 선발된 5명의 팀과 단지 직책만 보고 채용되거나 배치된 15명의 팀 중에서 어느 팀이 더 나은 결과를 낼까? 이와 같은 생산성 요인을 고려하면 비용 절감 효과와 같은 것은 순식간에 무색해진다. 또한 물가가 매우 높은 스톡홀름에 사는 최고의 엔지니어를 낮은 비용으로 모셔올 수도 있다.

이외에도 중요한 역할을 하는 다른 요소가 있지만 모든 것은 제대로 된 제품 팀으로부터 시작된다는 것, 그리고 제품 팀을 아웃소싱해야 한다면 적절한 근거로 그렇게 해야지, 단순히 비용 몇 푼 아끼자고 팀을 구성하면 안 된다는 것이 나의 굳은 믿음이다.

엔지니어링 팀이 재작성을 원한다면?!

"더 이상 새로운 기능은 없습니다! 이제 재작성해야 합니다! 지금의 코드 베이스는 골칫덩어리일 뿐이고, 사용자를 따라가지도 못하는 빈껍데기입니다. 더 이상 관리할 수도 없으니 그냥 그대로 내버려둬야 합니다." 엔지니어링 팀의 이런 말보다 제품 관리자가 무서워하는 것도 없다.

이런 상황을 겪은 회사는 많이 있었고, 지금도 곳곳에서 벌어지고 있다. 1999년 이베이가 그랬고, 당시 회사는 사람들의 예상을 뛰어 넘어 붕괴 직전까지 갔었다. 몇 년 전에는 프렌드스터(Friendster)에서도 벌어졌는데, 당시 프렌드스터는 마이스페이스(MySpace)와 연동하여 소셜 네트워킹을 접수하고자 했다. 그리고 마이크로소프트와 브라우저 전쟁을 한창 벌이던 넷스케이프도 피해가지 못했다. 그 결과 브라우저 전쟁에서 어느 쪽이 이겼는지는 따로 언급할 필요도 없을 것이다. 그런 상황이 벌어지면 대부분의 회사는 회복 불능 상태가 된다.

어떤 회사가 그런 상황에 맞닥뜨리면 대개 엔지니어링 팀을 비난한다. 하지만 개인적 경험에 비춰보면 일반적으로 제품 관리에 흠결이 있다는 것이 가혹하지만 부정할 수 없는 진실이다. 왜냐하면 그런 상황이 벌어졌다는 것은 대개 제품 관리자가 엔지니어링 조직을 몇 년씩이나 몰아붙이며 그들이 만들어 낼 수 있는 기능을 이미 죄다 쥐어 짜냈다는 의미이기 때문이다. 그렇게 된 것이라면 어느 시점에선가 인프라를 무시하게 되고 모든 소프트웨어는 필요한 기능을 더 이상 지원하지 못하는 지경까지 이르게 마련이다.

이와 같은 재작성이 진행되는 동안에는 고객이 예상했던 진전은 중단될 수밖에 없다. 재작성이 불과 몇 달 안에(또는 그보다 더 짧은 기간 안에) 이뤄질 수도 있다고 생각할지 모르지만, 어떤 경우든 그보다는 훨씬 더 오래 걸린다. 그리고 그동안 고객이 경쟁사로 이동하는 모습을 허망하게 바라보아야만 한다. 경쟁사는 그 와중에도 제품을 계속 개선하고 있는데도 말이다.

이런 상황을 아직 맞닥뜨리지 않았다면, 앞으로도 계속 맞닥뜨리지 않도록 방법 하나를 소개하고자 한다.

엔지니어링 팀의 능력을 쪼개어 이베이에서 통용되는 이른바 "헤드룸"에 할애하는 것이다. 빠르게 성장하면서 부딪히는 문제들 다수가 규모와 관련되기 때문에 한계에 다다르는 것을 피하자는 것이 헤드룸의 바탕 개념이며, 사용자 베이스의 성장, 트랜잭션의 성장, 기능의 성장을 위한 여지를 만들어 제품의 인프라가 조직의 니즈를 만

족시킬 수 있는 상태로 유지하는 것이 본질적인 방법이다.

엔지니어링을 구성할 때는 이렇게 해야 한다. 제품 관리에서 전체 능력 가운데 20%를 엔지니어링에 할애하여 그들이 원하는 곳에 쓰도록 한다. 할애된 능력은 주로 코드 베이스의 재작성, 리아키텍트, 리팩토링에 활용되거나 데이터베이스 관리 시스템의 교체, 시스템 성능의 향상 등, "이제 그만 중단하고 재작성해야 합니다."라는 말이 나오지 않도록 하는 데 필요한 곳이면 어디든지 활용되도록 한다.

지금 정말로 안 좋은 상황에 놓여 있다면 20%를 30%나 그 이상으로 늘릴 수도 있다. 하지만 20%보다 낮추면 잘 해내지 못할 가능성이 커진다.

만일 앞서 말한 상황에 처했다면 회사가 살아남지 못할지도 모른다. 그러나 회복할 수 있는 가능성을 얻고자 한다면 다음 단계들을 고려해 볼 만하다.

1단계 현실적으로 일정을 짜고 시간을 관리하여 엔지니어링 팀이 파악할 수 있는 필요한 변화를 이끌어내라. 경험 많은 엔지니어링 팀이라면 일반적인 개발 프로젝트가 진행되는 동안 꽤 정확한 예측을 쏟아내기 마련이다. 예외가 있다면 재작성의 경우다. 재작성에 대해서는 대개 낙관적으로 예측하는데, 이는 주로 진정한 재작성을 실제로 경험한 팀이 거의 없어서다. 이런 상황에서는 결정 내용을 반드시 엔지니어링 팀에 알려야 한다. 그래야 현실적인 날짜로 일정을 맞출 수 있다.

2단계 사용자가 사이트에서 제품 개발 진행 과정을 순차적으로 볼 수 있을 만큼 전체 재작성을 일정 부분으로 나눌 수 있다면, 반드시 그렇게 해야 한다. 재작성 기간이 9개월에서 2년 이상으로 늘어나더라도 사용자가 볼 수 있는 기능을 계속해서 구현하는(원래 기능의 25%에서 50%만 구현하더라도) 방법을 찾을 수 있다면, 이 방법은 해당 제품이 시장에서, 특히 빠르게 변화하는 인터넷 공간에서 힘을 발휘하는 데 믿을 수 없을 정도로 중요한 도구가 된다.

3단계 사용자가 볼 수 있는 기능을 전달하는 능력이 매우 한정되어 있기 때문에 그 가운데에서 적절한 기능을 골라내어 적절하게 정의했는지 확인해야 한다.

이베이는 거의 죽을 뻔 하고나서야 다시는 같은 위험에 빠지지 않을 수 있었다. 다시 말해 재작성을 해야 할 상황에 다시 처했을 때는 제반 문제를 제대로 처리한 것이다. 사실 이베이는 매우 빠르게 성장했기 때문에 3번이나 재작성을 해야 했는데, 3번째에는 전체 사이트를 다른 프로그래밍 언어와 아키텍처로 개편해야 했다. 또한 코드가 수백만 행에 이르는 광범위한 재작성을 몇 년에 걸쳐 진행하는 한편, 엄청나게 많은 새로운 기능을 무엇보다 사용자 베이스에 영향을 미치지 않고 전달해야 했

다. 나의 기억으로는 이베이가 가장 인상적인 엔진 재작성의 예가 아닐까 한다.

지금까지 언급한 상황을 타개하기 위한 최고의 전략은 그 지경까지 되지 않도록 하는 것이다. 세금을 납부하고, 헤드룸에 적어도 20%를 할애해야 한다. 아직 엔지니어링 담당자와 이런 논의를 거치지 않았다면 지금 당장 하라.