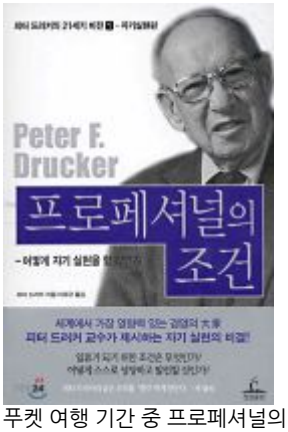
소프트웨어 개발자는 늘어나는데 생산성이 높아지지 않는 이유는?

> Spotify의 조직 문화 글을 읽다가 2010년에 조직 구조와 관련해 쓴 글이 있어 가져와 본다. Spotify의 조직 문화 글에서 다룬 Spotify가 만든 구조와 비슷한 부분이 있고, 내가 해결책을 찾지 못했던 부분도 있기 때문이다. 국내에서 Spotify와 같은 조직 구조와 문화를 같이 만들어 가고 싶지만 생각보다 쉽지 않다. 하지만 소프트웨어 개발에 있어 이 같은 요구와 도전은 계속해서 발생할 것이라 생각한다.



푸켓 여행 기간 중 프로페셔널의 조건(자기 실현편)이라는 책을 읽었다. 책을 읽기 시작할 때부터 나의 관심을 끄는 내용으로 시작하더니 끝까지 흥미진진하게 읽을 수 있었다. 현재의 나의 모습을 되돌아 볼 수 있는 기회도 제공해 주면서 앞으로 내가 나아갈 방향으로 제시해 주고 있다. 특히 지식 근로자의 생산성에 대하여 다루는 부분이 특히 좋았다.

내가 몸담고 있는 소프트웨어 개발자 또한 지식 근로자 중의 한 분야이기 때문에 더 많은 부분에서 공감할 수 있었다. 지식 근로자의 생산성과 앞으로 나가야할 방향에 대하여 많은 숙제를 안고 휴가에서 복귀한 시점에 CSO님이 사내 관리자를 대상으로 진행했던 강의 동영상을 볼 수 있었다. 강의를 듣다가 CSO님이 던진 질문의 원인과 해결책에 대한 내 나름대로의 생각을 정리하기 위하여 지금까지 고민했던 내용을 적어본다.

CSO님이 던진 질문은 다음과 같았다.

"개발자가 부족하다고 해서 몇 배의 개발자를 충원했다. 그런데 개발자 리소스는 항상 부족하다고 한다. 개발자가 충원되었음에도 불구하고 개발 생산성은 높아지지 않은 것 같다."

> 먼저 위 질문에서 개발자에 대한 정의가 필요할 것으로 생각한다. 내가 보는 관점에서 위에서 이야기한 개발자는 우리들이 흔히 이야기하는 프로그래머를 의미하는 것이 아니라 프로젝트에 참여하는 프로젝트 구성원으로 보는 것이 타당할 것으로 생각한다. 해외의 경우에는 프로젝트에 참여하는 구성원의 대부분이 개발자일 가능성이 높겠지만 국내 웹 서비스 개발 회사는 많은 업무로 나뉘어 개발이 진행되고 있기 때문에 이들을 통틀어 개발자로 보는 것이 맞을 것으로 생각한다.

이 질문에 대하여 지금 몸담고 조직에 있으면서 느꼈던 내용을 바탕으로 내가 생각하는 방향을 정리해보고자 한다. 물론 내가 지금 회사를 경영하는 경영자는 아니지만 효율적인 조직을 만드는 방법에 대해서 많은 관심을 가지고 있었기 때문에 내 나름대로의 원인과 해결책을 가지고 살아왔다. 몇 번의 포스팅을 통하여 이 내용을 다루어 보도록 하겠다. 이 글에서 다루고 있는 내용이 내가 몸담고 있는 조직에 국한된 것이 아니라 소프트웨어 개발을 하고 있는 많은 조직에서 발생하고 있는 현상이리라 생각한다. 어쩌면 소프트웨어 개발 조직 뿐만 아니라 앞으로 점점 지식 근로자가 많아지는 회사에서 경험하게 될 현상이 될 수도 있다고 생각한다.

자 그럼 이제 본격적으로 이야기를 시작해보자.

소프트웨어 개발은 다른 분야에 비하여 역사가 길지 않기 때문에 생산성 향상을 위한 방법을 다른 분야의 성공 사례를 도입하여 적용하는 것이 현실이다. 그 중 대표적인 예가 제조업의 경험으로부터 분업화와 자동화를 도입하려는 시도가 많았다. 또한 제조업, 건축 분야로부터 모듈화를 통한 재사용 가능한 컴포넌트화에 대한 시도가 그것이리라. 이 글에서는 자동화와 모듈화를 통한 생산성 향상에 대해서는 언급하지 않았다. 이 글에서는 제조업 분야로부터 도입한 분업화에 대하여 언급하고자 한다.

프로페셔널의 조건에 나오는 내용을 보면 다음과 같은 내용이 나온다.

> 작업은 연구될 수 있고 분석될 수 있으며, 또한 작업은 일련의 간단하고도 반복적인 동작으로 나눌 수 있다는 것 - 즉 작업에서의 각 동작은 하나의 옳은 방법으로, 주어진 시간 내에, 알맞은 도구를 사용하여 수행될 수 있다. - 테일러의 주장
>
> 테일러의 주장에 의한 훈련 방식을 도입한 국가들의 생산력은 거의 50배 가까이 증가하였는데, 이 놀라운 생산력 증대야말로 모든 선진 국가에서 생활 수준과 삶의 질을 월들히 향상시킬 수 있었던 근원이었다.
>
> - 피터 드러커, 프로페셔널의 조건 1장 일부 발췌 및 정리

위 인용문의 테일러의 주장은 한마디로 말해서 모든 작업을 분석하여 세분함으로써, 단순화할 수 있고, 짧은 기간의 훈련만으로 '일류 기술자'를 만들어 낼 수 있다는 것이다. 이 주장은 제조업 분야에는 적중했으며, 이로 인해 생산성 혁명이 가능하게 되었다.

이와 같이 제조업에서의 비약적인 생산성 향상은 새롭게 등장한 소프트웨어 개발 분야에서도 매력적일 수 밖에 없었다. 하지만 제조업에서 성공적이었던 이 방법이 소프트웨어 개발에 있어서는 성공하지 못하고 있는 듯하다. 그 이유에 대하여 Pete McBreen은 다음과 같이 이야기하고

있다.

> 소프트웨어 개발에 있어서 작업을 세분화하고 분업화할 때 작업이 더 작은 단계로 세분화될수록, 한 사람에서 또 다른 사람으로 정보를 전달하는 데에 더 많은 시간이 걸린다. 생산라인 접근 방식은 수작업 노동에는 잘 맞을 수 있다. 그러나 지적인 작업에는 형편없이 실패한다.
>
> 소프트웨어 개발은 팀원들의 머리에서 일어난다. 사람들을 특정 활동에 전문화시킴으로써 간단한 프로젝트의 딜리버리도 여러 경로를 통해야 한다. 각 경로는 실수와 결함에의 잠재성을 갖고 있는 값비싼 과정이다.
> - Pete McBreen, 소프트웨어 장인 정신

소프트웨어 개발에 있어서 작업을 세분화할 경우의 문제점에 대하여 설계 작업과 개발 작업을 분리할 경우 발생할 수 있는 한계점을 다루고 있는 다음 글을 읽어보면 많은 부분에서 공감할 수 있을 것이다.

* [설계의 본질, 그리고 UML - 1부|http://aeternum.egloos.com/1545662]
* [설계의 본질, 그리고 UML - 2부|http://aeternum.egloos.com/1561326]
* [설계의 본질, 그리고 UML - 3부|http://aeternum.egloos.com/1576712]

소프트웨어 개발에 있어 세분화, 분업화로 인하여 얻을 수 있는 한계가 있다는 것을 알고 있지만 아직도 많은 소프트웨어 개발회사는 제조업의 생산성 향상 방식을 도입하고 있다. 이와 같이 변화하게 되는 과정을 살펴보면 다음과 같다.

회사가 성장하는 초기 단계에는 각 서비스(또는 프로젝트)별로 업무를 세분화하여 프로젝트를 진행한다. 물론 이 시점에 업무가 세분화되어 있지만 문제가 될 정도로 세분화되어 있는 상태는 아니다. 조직 구조 또한 서비스(또는 프로젝트)를 기준으로 분리되어 있는 경우가 많다. 회사가 성장하는 단계에 있어서 일정 수준의 분업화와 조직 구조는 크게 문제가 되지 않는다. 하지만 회사가 성장하고 조직이 커지면서 전문화와 효율화를 해야 한다는 이유로 각 업무별로 조직 구조가 바뀌고, 업무는 점점 더 세분화되어 간다. 이 같은 논리는 다분히 과거의 제조업에서의 생산성 향상에 대한 경험에 기반을 두고 있는 것으로 판단된다.

예를 들어 초기 단계에서는 다음 그림과 같이 업무 분석가(또는 기획자), 개발자, 디자이너, DBA(선택) 정도로 시작하는 경우가 많다.

시간이 지나면서 서비스가 성장하고 회사의 규모가 커질수록 각 프로젝트에 참여하는 역할은 세분화된다. 이에 대한 요구는 각각의 업무에 대한 전문성을 강화하기 위한 일환으로 진행된다. 전문화함으로써 소프트웨어의 품질을 향상시킬 수 있으며, 생산성 또한 높아질 것이라는 판단 때문이다.

이와 같이 프로젝트 하나를 진행하기 위하여 여러 개의 업무로 나누어지고, 각각의 업무를 담당하는 구성원들은 점점 더 증가한다. 시간이 지나면서 각각의 업무 담당자들은 자신과 같은 업무를 하고 있는 동료와 같은 조직에서 일하는 것이 자신의 실력을 향상할 수 있으며, 전문성을 더 강화할 것이라 생각한다. 이 같은 생각이 확대되면서 자연스럽게 각 업무는 하나의 조직으로 만들어지게 된다. 이와 같이 각각의 업무가 조직으로 바뀌게 되면서 프로젝트를 진행하기 위하여 관여하는 구성원이 한 개인이 아니라 조직(대표적으로 팀)이 되어버리는 상황이 발생한다.

많은 개발자를 충원했지만 개발 생산성이 향상되지 않는 가장 큰 원인에 대하여 내가 내린 결론은 앞에서 본 바와 같이 업무 단위를 너무 세분화, 분업화했기 때문이다. 업무 단위를 세분화하면서 조직 구조가 각 서비스(또는 프로젝트) 단위로 구성되었다면 좀 더 효율적이었을 것이라 생각한다. 하지만 조직 구조까지 업무 단위로 재편되면서 개발 생산성은 급격하게 떨어지게 된다. 업무 단위가 조직 구조로 변경되는 초기 단계에는 기존에 진행하던 업무 방식이 남아 있기 때문에 큰 변화는 없거나 오히려 각 업무 단위가 전문화 되면서 생산성이 향상된다는 착각을 할 수 있다. 하지만 더 큰 문제는 업무 단위 기반의 조직이 점점 더 탄탄해 지면서 발생하게 된다.

이번 글에서는 소프트웨어 개발 회사에서 개발자는 지속적으로 충원되지만 개발 생산성이 높아지지 않는 이유에 대하여 나의 생각을 정리해봤다.다음 글에서는 업무 단위를 기반으로한 조직이 점점 더 탄탄해지면서 발생할 수 있는 문제점에 대하여 살펴보도록 하겠다.

PS. 개발자가 늘어남으로 인해서 생산성이 떨어지는 이유는 이 이외에도 무수히 많을 것으로 생각한다. 이 글에서 내가 선택한 원인은 여러 개의 원인 중에서 가장 큰 원인이라고 생각하는 부분에 대하여 다루었다. 이후에 기회가 된다면 더 많은 원인에 대하여 이야기해보고 싶다.

업무, 조직 세분화로 인해 발생하는 문제점은?

> 이 글은 내가 포털 회사에 4년 간 몸담고 있으면서 느꼈던 내 나름대로의 생각을 정리한 것이다. 이 글의 내용은 포털과 같이 하나의 소프트웨어를 지속적으로 관리하고 성장시켜 나가야하는 소프트웨어 개발 회사에 도움이 될 듯하다. 나는 지금까지 웹 에이전시, SI 회사에 몸담은 경험이 더 많은데 웹 에이전시, SI와 같이 특정 회사에 업무 의뢰를 받아 단기간에 개발을 완료해야하는 조직에는 적합하지 않을 수 있다.
>
> 이 글의 내용은 내가 몸담고 있는 조직의 의견과는 무관하다. 지금까지 소프트웨어 개발 업무를 진행하면서 느꼈던 내 나름대로의 생각을 정리한 것이다. 내가 이 글을 쓰는 가장 큰 이유는 내가 잘못 판단하고 있는 부분도 많을 것이기 때문에 그에 대한 의견을 받아보고 더 바람직한 모습에 대한 논의를 해보고자 함이다.
>
> 논리적으로 타당하지 않거나 근거가 부족하다고 생각하는 부분에 대해서는 가차없이 의견 마구마구 날려주기 바란다.