린 스타트업을 읽고서...

린 스타트업을 읽은 후 어딘가에는 몇 자 끄적여 놓아야할 듯 해서 글을 남긴다. 린 스타트업을 읽으면서 현재 내가 운영하면서 점진적으로 개발하고 있는 slipp.net 커뮤니티에 대해 많은 것을 고민할 수 있는 계기가 되었다. slipp.net을 처음 시작할 때의 마음은 핵심 기능만을 가지는 방향으로 개발하고 일단 오픈하는 것이었다. 그리고 사용자의 피드백을 받아서 점진적으로 개선해 나가는 것이었다. 그런데 서비스를 일단 오픈하고보니 무엇을 개선해야 될 것인지 찾기가 쉽지 않았다. 그래서 개인적으로 필요하다고 생각하는 기능들을 사용자 입장에서 하나씩 추가해 나가고 있었다.

그런데 린 스타트업 책을 읽다보니 내가 얼마나 주먹구구식으로 개발하고 있는지를 느낄 수 있었다. 정말 생각 없이 그냥 내가 필요하다고 생각하는 기능들만 추가하는데 집중했구나. 가치를 주는 기능인지도 모르면서 그냥 열심히 개발하는데만 집중하고 있었다라는 생각이 들었다. 이 책에서 제시하고 있듯이 가설을 세우고, 만들고, 학습을 통해 점진적으로 발전시켜 나가야겠구나라는 생각을 하게 되었다.

이 책을 읽다가 박영록씨가 쓴 Quick And Dirty라는 글을 읽게 되었다. 이 글 또한 그 동안 내가 고민하고 있던 많은 부분에 대한 해결책을 제시해 주는 듯하다. 특히 이 글의 린스타트업과 TDD를 비교하는 부분이 마음에 든다. 그 부분을 인용해 보자.

> 린 스타트업은 말하자면 TDD를 조금 더 큰 스케일로 하는 것이다.
>
> 1. 가설을 수립하고, 가설을 확인할 수 있는 지표를 설정한다.
> 2. 가설을 검증할 수 있는 MVP를 만든다. quick & dirty!
> 3. 가설이 검증되었다면 완성도를 높여간다.
> 4. 가설 검증 결과가 틀린 것으로 나온다면 새로운 가설을 수립한다.

얼마 전 next 한 학생으로부터 다음과 같은 이야기를 들었다. 스타트업에서 여유롭게 테스트 코드를 만들 수 있나요? 최대한 빨리 만들어서 배포하고 사용자에게 서비스를 제공하는 것이 최우선인데요. 맞다. 나도 일정 부분 공감한다. 하지만 아무리 quick & dirty를 추구하더라도 일정 시간이 지나면 부채를 갚아야 한다. 그렇지 않을 경우 결국에는 부채의 늪에 빠져 처음과 같은 속도를 낼 수 없을 것이다.

린 스타트업의 234~235페이지를 보면 비슷한 이야기가 나온다. 부채를 갚지 않아 실패한 경험담을 공유하고 있다. 특히 국내 수 많은 소프트웨어가 초반에는 성공하는 듯 보이다가 일정 시간이 지나면서 실패한다. 그 이면에는 부채를 갚지 않고 쌓아둔 결과 사용자의 요구에 발빠르게 대응하지 못해 외면을 당하는 경우가 많을 것이다. 최근에 수 많은 스타트업이 세워지고 있다. 스타트업이라고 quick & dirty만 추구하지 말고 위 과정에서도 보듯이 가설이 검증되면 완성도를 높여가는 과정에서 단위 테스트 코드도 만들고 리팩토링도 하면서 발전해 가는 것이 더 빠른 길이라 생각한다.

내가 스타트업을 만들고 개발한 경험이 없지만(slipp.net도 그 범주의 일환이라고 본다면 비슷한 부분이 있다고 생각한다.) 다음과 같은 방식으로 접근하면 어떨까라는 생각을 해본다. 스타트업을 시작하는 시점에서 단계적으로 살펴보자. 내가 직접 경험한 것은 아니고 책을 읽으면서, 지금까지 프로젝트를 진행하면서, slipp.net을 운영하면서 느낀 점을 정리했다.

- 앞의 인용문처럼 가설을 수립하고, 가설을 확인할 수 있는 지표를 설정한다.
- 가설을 검증할 수 있는 Minimum Viable Product(MVP)를 만든다. quick & dirty

대부분의 스타트업들은 MVP를 최대한 빨리 만들어 가설을 검증하려고 할 것이다. 그러면 MVP를 배포한 후에 무엇을 하는 것이 좋을까? 위 3단계에 해당하는 가설이 검증되는 단계는 MVP를 사용자에게 배포한 후에 바로 가능하지 않을 것이다. 최소 한 달에서 두 달의 시간은 기다린 후에야 가설을 검증할 수 있는 데이터가 쌓이리라 생각한다. 이 기간은 앞에서 MVP를 만들 때보다는 시간적인 여유가 있으리라 생각한다.가설이 검증되지 않은 이 기간에 새로운 기능을 추가하려고 노력하기 보다는 지속적인 빌드와 배포 환경을 구축하는데 시간을 쏟는 것이 바람직할 것으로 본다. Quick And Dirty 글에서도 이야기하듯이 MVP가 가설을 검증하기 위한 최소한의 기능만 가지고 있다면 부채를 충분히 감당할 수 있는 수준이라 생각한다. 아직 가설이 검증된 단계가 아니기 때문에 테스트 코드를 만들고 리팩토링을 하는 것이 의미 없는 투자가 될 수도 있다. 가설이 검증된 후 테스트 코드를 만들고(가능하면 TDD로) 리팩토링하는 해도 늦지 않을 것이다.

하지만 지속적인 빌드와 배포에 대한 투자는 가치있는 작업이라 생각한다. 지속적인 빌드와 배포는 첫 번째 가설을 검증하는데만 활용할 것이 아니라 첫 번째 가설이 틀렸을 경우 두 번째 가설을 검증하기 위해 새로운 기능을 배포하는 시점에도 유용하게 사용할 수 있을 것이다. 처음부터 거창할 필요는 없을 것이다. 지금 시점에 지속적인 빌드와 배포가 가능한 수준으로 프로젝트에 원칙을 세워 소스 코드를 관리하고 빌드를 개선하고 배포를 자동화하는 것이다. 이 모든 과정에는 학습 과정이 필요하다. 즉, 시간이 필요하다는 것이다. 이 학습 과정은 뒤로 미루면 미룰 수록 더 큰 부채로 다가오는 경우가 많다. 따라서 가능한 빠른 시점에 프로젝트 전체 참여자가 빌드와 배포 관련한 지식을 가질 수 있도록 하는 것이 중요하다고 생각한다.

린 스타트업 책에서도 에릭 리스는 지속적인 빌드, 배포, 통합에 대해서 강조하고 있다. 가설이 검증되지 않은 상태에서 많은 기능을 추가하는데 시간을 투자하기 보다는 서비스를 운영하기 위해 자동화가 필요한 부분을 최대한 자동화하고 서로 간의 협업을 위해 필요한 원칙을 세우고 학습하는데 시간을 투자하는 것이 스타트업이 성공할 수 있는 지름길이지 않을까 생각한다.

스타트업에 경험이 없는 개발자의 주장이라 얼마나 신뢰가 갈지는 모르겠지만 내가 생각하는 모습을 정리해 봤다. 어떤 선택을 할 것인지는

스타트업을 하는 회사의 몫이리라.