Quick and Dirty에 대한 지나친 환상을 버려라.

이 글을 쓰게 된 계기는 최근 페이스북에 올라온 글(https://www.facebook.com/patternloader/posts/10201066888494375)을 본 이후이다. 분명히 맞는 측면도 있지만 초보 개발자가 오해할 가능성이 있어 이 글을 보완하자는 관점에서 글을 쓴다. 페이스북에 올라온 글을 다시 한번 살펴보자.

```
    ,    ,          .

 DevOps          ,

              .   .       ,   ?

  TDD ? QA   ?          . Pollution  0     ?   ..


Dirty and Quick  ..   ..     .
```

나도 이 글에서 이야기하는 "사용할 가치가 있는 소프트웨어를 만들자."는 의견에 100% 공감한다. 소프트웨어를 만드는 개발자들이 반드시 가져야하는 마음가짐이다(알고는 있지만 기술도 이해하고 사람도 이해해야지라 정말 힘들다). 하지만 관리자나 초보 개발자가 Quick and Dirty에 대한 환상을 가질 가능성이 있다고 생각한다. 최근 린스타트업 책에서도 MVP를 강조하고 있다. 이 책을 읽으면서 상당수의 사람들은 Quick을 생각했을 것이다. 약간의 품질은 희생해도 된다고 생각할 가능성도 높다. 사실 이 책에서는 기술적인 품질 보다는 고객의 관점에서 소프트웨어의 품질을 강조하고 있기 때문에 기술적인 관점에서의 품질은 다루지 않고 있다.

나를 포함해 대부분의 사람들의 심리는 자신이 듣고 싶은, 읽고 싶은 것만 듣고, 받아 들이려는 경향이 높다. 또한 자신의 관점으로 이해하고 해석하려는 경향이 있다. 따라서 기술적인 품질에 대해 깊이 있는 고민을 해보지 않았거나 경험이 없는 상당 수의 개발자는 린스타트업 책을 읽으면서 기술적인 품질은 그리 중요하지 않다고 생각할 수 있다.

먼저 Quick and Dirty에 대해 좀 더 깊이 있게 다루고 있는 QuickAndDirty 글을 먼저 읽어봤으면 좋겠다. 이 글을 보면 Quick And Dirty에 대한 다양한 관점을 정말 잘 정리하고 있다. 특히 프로그래밍을 구현할 때 Quick And Dirty 방식으로 개발하는 방법에 대해서까지 구체적으로 다루고 있다. Quick And Dirty에 대한 자신만의 관점을 만들고 싶다면 한번 정독해볼 것을 권한다. 이 글의 일부분을 인용해보자.

> clean code에 도달하는 가장 훌륭한 방법인 TDD의 사이클을 복습해보자.
>
> 1. 제대로 동작하는지 확인할 수 있는 테스트를 작성한다.
> 2. 실행 가능하게 만든다. 다른 무엇보다도 중요한 것은 빨리 초록 막대를 보는 것이다. 빨리 초록 막대를 보는 것은 모든 죄를 사해준다. 하지만 아주 잠시동안만이다.
> 3. 올바르게 만든다. 이제 시스템이 작동하므로 직전에 저질렀던 죄악을 수습하자.
>
> 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.
>
> 문제는 바로 2와 3 사이에 있다. 린 스타트업은 개발자를 위한 책이 아니라 스타트업을 위한 일반적인 방법론이므로, 2와 3 사이에 TDD 사이클의 3번 과정을 따로 시간을 할애해서 적어놓지 않은 것이다. 가설 검증에 성공한 스타트업이 이 2.5 리팩토링을 놓치게 되면 바로 이어 등장하는 추격자들에게 자리를 내주게 된다. 추격자들은 모범 답안이 있으니 빠른 속도로 MVP를 만들 게 아니라, 처음부터 완성품을 보고 달려도 되므로, 완성품에 더 빨리 도달할 수 있다. MVP까지는 quick & dirty로 빠르게 만들 수 있지만, 완성품을 만드는 것은 clean code 없이 속도를 낼 수 없다.
>
> 하지만, 설령 개발자가 2와 3 사이에 숨겨진 2.5를 인지한다고 해도 문제는 간단하지 않다. CEO는 quick & dirty로 가설을 검증해냈기 때문에 이미 quick & dirty에 맹신이 생긴 상태이고, 앞으로도 쭉 quick & dirty로 해나가면 된다고 생각한다. 그래서, 계속 같은 속도로 움직일 수 있을 거라고 생각한다. 이런 CEO에게 코드를 정리할 시간이 필요하다고 말하는 것이 먹힐 가능성은 매우 낮다. 설령 말로는 동의해도 이후에 액션들은 2.5 작업을 할 수 있도록 내버려 두지 않는다. proof of concept 이후에도 쭉쭉 뻗어나가는 스타트업과, 경쟁에 뒤쳐지는 스타트업에는 결정적인 차이가 바로 이것이다.

위 글이 Quick and Dirty 방식으로 접근할 때 가장 중요한 부분이라고 생각한다. Quick and Dirty의 본질은 잠시 동안 죄악을 저질러도 묵인한 후에 가설에 대한 검증이 완료되면 완성도를 높여가야 한다. 완성도를 높이는 작업 중의 대부분은 기술적인 부채를 갚아야 하는 것이다.

국내 소프트웨어 개발은 지금까지 어떤 방식으로 일을 해 왔을까? 나는 국내 대부분의 소프트웨어 개발이 Quick and Dirty 전략으로 지금까지 발전해 왔다고 생각한다. 하지만 좋은 소프트웨어가 나오기 힘든 원인 중의 하나가 Quick and Dirty 후에 Clean이라는 과정이 없었기 때문이다. 지금까지 국내 소프트웨어 개발은 Quick and Dirty 후에 다시 Dirty만 있었지 Clean이라는 과정이 없었다. 이런 과정으로 개발하기 때문에 신선한 서비스들은 많이 만들어 낼 수 있었지만 지속 가능한 서비스를 만드는데 한계가 있었던 것이 사실이다. 국내 소프트웨어의 수명이 짧은 이유 중의 하나일 것이다.

그렇다면 Quick and Dirty 작업을 한 후 Dirty가 아니라 Clean하는 작업을 하려면 어떤 경험을 쌓아야 할까? 어느 정도 수준의 개발자 경험일 때 가능할까? 나는 Qucik and Dirty 접근 방식은 초보 개발자나 소스 코드 품질 관리에 대한 경험과 노하우가 없는 경력 개발자가 접근할 경우 위험하다고 생각한다. 지금까지 국내 소프트웨어 개발 과정을 살펴보면 경험 없는 개발자들이 Quick and Dirty 접근 방식으로 개발했을 때의 결과는 보지 않아도 뻔하다. Quick and Dirty 접근 방식은 소스 코드 품질을 제대로 관리해본 경험있는 개발자가 시도했을 때 성공할 수 있다. CEO나 임원이

아무리 재촉해도 부채를 갚아야할 시점에는 부채를 갚는데 시간을 투자해야 한다고 용기있게 말하고 실행할 수 있는 개발자가 시도했을 때 성공할 수 있다. 물론 회사를 만드는 목적이 짧은 기간(1, 2년 정도라 생각한다.) 동안 참신한 서비스를 만들어 다른 회사에 매각하는 것을 목표로 한다면 기술적인 부채를 가진 상태로 유지할 수도 있을 것이다. 단지 이런 생각만으로 회사를 만들고 서비스를 만든다면 논할 가치도 없다고 생각한다.

Quick and Dirty는 우리가 기본적으로 지켜야하는 원칙을 깨고 응용하는 실천 방법이다. 그렇다면 응용의 단계는 어느 시점에나 가능할까? 먼저 기본적으로 지켜야할 원칙을 제대로 지키는 훈련이 되어 있어야 한다. 일정 수준 훈련 과정을 거친 후에 자신이 지키고 있는 원칙의 한계를 이해하고, 상황에 따라 유연하게 응용할 수 있는 단계가 된다. 따라서 Quick and Dirty 실천 방법을 사용하려면 먼저 제대로 된 소스 코드 품질 관리 경험이 있어야 한다. 기본 훈련이 잘 되어 있어야 응용할 시점을 알고, 감당할 수 있는 기술 부채에 대한 감(sense)도 생길 수 있다.

Quick and Dirty 방식으로 개발하고 싶다면 반드시 TDD는 아니더라도 단위 테스트를 만들고, 리팩토링을 하는 연습부터 하자. 소스 코드 품질을 관리하기 위한 다양한 활동들에 대해서 학습하자. clean code를 만드는 경험은 쉽게 쌓이지 않는다. 따라서 소스 코드 품질에 대해 항상 고민하고 훈련해야 한다. 1만 시간의 법칙이 맞다면 하루에 4시간씩 10년동안 의도적 수련을 해야 제대로 된 품질 관리를 할 수 있을지도 모른다.

그렇다고 무조건적으로 Quick and Dirty 방식으로 접근하지 말라는 것이 아니다. Quick and Dirty 방식으로 접근하되 항상 품질을 염두에 두고 접근해야 한다는 것이다. 그리고 품질에 대한 깊이 있는 경험이 있는 개발자가 한, 두명은 포함되어 있어서 감당할 수 있는 기술 부채의 한계를 알고, 품질을 높이는 시간을 확보할 때 Quick and Dirty가 성공할 수 있다는 것을 이야기하고 싶은 것이다.

Qucik and Dirty에 대한 지나친 환상을 버리자. Quick and Dirty에 대한 지나친 환상은 과거에도 그러했고, 현재도 그러한 (Quick and Dirty 후에 다시 Dirty하는) 대한민국 소프트웨어 개발 과정을 반복할 가능성이 높다. Quick and Dirty 후에는 반드시 Clean 과정이 포함되어야 하며, 이는 Clean에 대한 제대로 된 학습과 경험이 있는 개발자가 시도해야 성공할 수 있다.

PS. 단순히 돈을 버는 측면으로만 생각한다면 이 글의 이야기가 이상주의적이라 생각할 수도 있다. 하지만 돈 뿐만이 아니라 개발자라는 직업적인 관점에서도 살펴봐야하지 않을까? 나는 부채가 있는 소스 코드를 Clean하는 과정에서 엄청난 즐거움을 느낀다. 아마도 상당 수의 개발자가 이 과정에서 창의적인 경험을 하고, 즐거움을 느낄 수 있다. 우리가 Quick and Dirty에만 집중한다면 어쩌면 돈에만 집중하고 있을지도 모른다. 그 이면에 있는 프로그래머의 자존심과 즐거움은 무시되고 있는지도 모른다.

나는 결과보다는 과정에 집중하고 싶다. 과정에서 즐거움을 느낀다면 결과 또한 좋으리라 생각한다. 결과가 좋지 않더라도 후회하지 않으리라 생각한다. 최근의 사회적인 흐름이 과정보다 결과에 집중하는 경향이 강해져 안타까울 따름이다. 개발자들이 프로그래밍이라는 일이 정말 즐겁고, 보람있는 일이라 느끼게 하고 싶다면 결과 보다는 과정에 집중했으면 좋겠다.