김창준 애자일 컨설팅 대표

반가운 TDD 수련서

채수원님을 꽤 오래 전부터 한국XP사용자모임(XPer)을 통해 뵈어 왔습니다. 겸손하고 성실한 태도, 그리고 커뮤니티와 애자일에 대한 열정을 보면서 깊은 인상을 받았습니다. 이번에 TDD에 대한 책을 직접 저술하신다고 하여 이렇게 원고를 읽고 추천사를 쓰게 되었습니다. 무엇보다 이 책이 국내 TDD 확산에 큰 도움이 될 것이라고 생각을 합니다.

이 책의 장점을 몇 가지 짚어보고 싶습니다.

첫째, 입체적이고 다채롭습니다. 전문가 인터뷰에서 TDD에 대해 경험이 많은 분들의 경험담을 접할 수 있고, 테스트들을 어떤 식으로 구조화하고 정리하는 것이 좋은지에 대한 팁도 있으며, 개발 영역에 따른(웹이나 데이터베이스 같은) TDD 작성 패턴도 소개하고 있습니다. 아마도 많은 베타리더들과 함께 "돌멩이 수프"를 만들면서 가능했던 일이 아닐까 짐작해봅니다.

둘째, 여러 가지 도구를 구체적으로 다루고 있습니다. TDD를 하면서 사용하는 도구들이 참 다양합니다. 이 책에서는 이런 도구들을 사용 예를 들어가며 단계별로 방법을 안내하고 있습니다. 뭔가 추상적인 이야기를 늘어놓는 것보다 구체적으로 각 단계에서 무엇을 해야 하는지 알고 직접 부딪혀 보는 것을 선호하는(쉽게 말해 "백문이불여일타"를 금과옥조로 믿는) 학습자에게 큰 도움이 되리라 생각합니다. TDD에 대해 충분히 감이 안 온 분들이 책을 보고 따라하면서 바로 시작할 수 있다는 장점이 되지요.

셋째, 소개된 도구들이 비교적 최신의 것들이어서 책을 읽고 따로 인터넷의 최신 문서를 공부해야 한다든지 하는 불편함이 없습니다. TDD의 도구들도 많은 발전을 해왔습니다. 국내에 출간된 몇 안 되는 TDD 관련 책들 상당수가 이런 도구의 발전 속도를 따라오지 못하고 있다고 봅니다. 이 책에서는 (또 얼마 지나지 않아 새로운 도구들이 나오긴 하겠으나) 최신 도구를 많이 다루고 있습니다.

넷째, TDD를 배우는 사람 입장에서 가질 만한 궁금증들에 대한 답이 많습니다. 8장이나 9장이 특히 그렇습니다. 9장의 FAQ는 몇몇 분들에게 오아시스 같은 느낌이 들 것 같습니다. 좀 더 확장되었더라면 좋지 않았을까 하는 아쉬움이 있습니다.

이제는 이 책의 독자들에게 주의점을 한 가지 말씀드리도록 하겠습니다. 도구나 술기에 너무 매몰되지 말고 고민을 하면서 수련을 즐기시라는 것입니다. 이런 점을 주의하면서 이 책을 읽는다면 훨씬 더 많은 것을 얻으시리라 생각이 듭니다.

저는 10년이 넘게 전통무술을 수련해 오고 있습니다. 그 동안 무술을 배우려는 사람 중에 술과 형에 현혹된 사람들이 간혹 보였습니다. 술은 술기이고 형은 형식입니다. 모든 무술에는 술기와 형식이 있습니다. 그리고 거기에는 일종의 직선적 단계가 존재합니다. 널리 알려진 태권도로 예를 들자면, 태극 1장하고 나면 태극 2장하고, 계속 하다보면 고려, 금강, 십진 같은 것을 배우게 되겠죠.

여기에 유혹이 존재합니다. 십진 위에 뭔가가 또 있다고 소문이 돕니다. 문주가 직접 가르친다고 합니다. 그러면 모이는 사람들이 있습니다. 자연스러운 것이죠. 하지만 그 중에는 내공이 충분하지 못하고 수련이 안된 사람들도 있습니다. 자신의 수준이 별볼일 없지만 그 "뭔가"를 배우면 나의 내공도 같은 수준으로 높아질 것이라, 뭔가 대단한 사람이 될 것이라 생각합니다.

이 부분은 착각입니다. 형을 배운다고 전문가가 되지 않습니다. 전문가가 되기 위해서는 고민과 수련의 절대적 양이 필요합니다. 수많은 고민의 시간들을 그냥 건너뛰어버릴 수는 없습니다.

- 이것이 정말 최선일까? 더 나은 방법은 없을까?
- 무엇이 원칙이지? 그 원칙을 현 상황에 적용하려면 어떻게 해야 할까?
- 불가능해 보인다. 하지만 가능하다면 어떤 모양일까?
- 왜 이렇게 고통스러울까? 더 쉽게 할 수는 없을까? 아예 이런 문제를 해결할 필요가 없게 만들려면?

내공을 키우려면 이런 고민들을 수없이 해야 합니다. 반면에 어떤 도구를 사용해야 하지? 어떤 기법을 써야 하지? 같은 고민들은 비교적 형에 가까운 경우가 많습니다. 물론 내공이 깊어지면 이런 형에 대한 고민들도 수준이 같이 높아집니다. 하지만 진짜 문제는 내공을 수련하지 않으면서 처음부터 형에 집착하는 경우입니다. 앞에서 말한, "좀 더 높은 수준의 형을 배워야지"라는 욕심 말입니다.

한편 기존에 있는 도구의 사용이 학습에 짐이 되는 경우를 보기도 했습니다. 도구의 존재(특히 그 도구가 유일하다면)가 고민의 기회를 빼앗아 버리는 경우가 있습니다. 아, 이런 도구가 이미 있구나. 이대로 쓰면 되는 거구나. 또 도구의 복잡성이 고민의 여유를 빼앗아 버리는 경

우도 있습니다. 이 도구를 어떻게 써야 할지 배우는 것 자체가 또 하나의 부담이 되어서 원래 문제를 고민할 여유가 줄어드는 것이죠(이런 것을 인지부하이론Cognitive Load Theory이라고 합니다 -새로운 수학 이론을 외국어로 배운다고 생각해 보면 감이 쉽게 올 겁니다).

TDD에서 좀 구체적인 예를 들어보죠.

- DbUnit이 최선일까? 장점은 무엇이고, 단점은 무엇일까?
- 나는 왜 DbUnit을 쓸까? 그 이유를 위해서 꼭 DbUnit이어야 하는가? 더 나은 방법은 없나?
- 안 쓰고 어떻게 할 수 있을까? 안 써서 더 나은 점이 있을까? 어떨 때, 어떤 방식으로 쓰면 좋을까?
- 데이터베이스 관련 개발을 할 때 어떤 원칙이 중요한가? 그 원칙을 어떻게 적용할까?
- 멀티스레드 프로그래밍을 TDD로 쉽게 하는 방법이 존재한다면, 그 방법이 무엇일까?
- 멀티스레드를 어떻게 테스트하느냐의 문제 자체를 피하는 방법은 뭘까?
- 소득세를 계산하는 코드와 그 코드에 대한 테스트 코드(부록 참고)에서 내가 못 본 차원의 중복이 존재한다면? 어떻게 리팩토링할까? 그걸 리팩토링한 다음에 또 중복이 있다면 뭘까? 언제까지 리팩토링할 수 있을까?

이런 고민을 하는 것이 절대 쉽지는 않습니다. 만족스러운 답을 구하지 못할 수도 있습니다. 하지만 그 과정에서 많은 공부와 수련이 되고 내공이 쌓이게 됩니다. 그 여정에 도움이 될만한 팁을 드린다면, 실행전략뿐 아니라 학습전략에 대해서도 생각을 해보라는 점입니다. 인지적 부하를 낮춘 상태에서 고민 그 자체에 몰두할 수 있는 환경을 만들어 여러 번 안전한 실험을 해보세요. 예컨데 멀티스레드에 대한 고민을 하려면 가장 간단한 멀티스레드 코드와 개발환경을 갖추고(각종 도구나 기법 등의 변수를 제거하고) 해보시라는 겁니다.

항상 덜 고통스럽고 유쾌한 길이 존재한다는 믿음을 가지고, TDD 수련을 즐겨보시기 바랍니다. 가시는 길에 진달래가 피었거든 가만히 앉아서 구경도 하시면서 말이죠. 꽃 구경할 시간 없다고요? 뭐가 그리 바쁘십니까! 아, 이 책도 챙겨가세요. 그 여정에 이 책이 좋은 길동무가 될 겁니다.