실용주의 프로그래머

1. 요구사항의 구렁텅이

자신이 뭘 원하는지 정확히 아는 사람은 아무도 없다.

프로그래머의 일은 사람들이 자신이 원하는 바를 깨닫도록 돕는 것

이것이 우리의 가치가 가장 빛나는 부분이다.



2. 요구사항은 과정이다.

요구사항은 피드백을 반복하며 알게된다. 의뢰인에게 피드백을 주고, 의뢰인은 피드백을 바탕으로 자기 생각을 더 가다듬는다.

실용주의 프로그래머는 프로토타입을 만듦

프로젝트 전체를 요구사항 수집 과정으로 보아야 하며, 반복 주기가 끝날 때마다 직접 의뢰인에게 피드백을 받는 형태로 궤도에서 벗어나지 않을 수 있음

3. 의뢰인의 입장에서

의뢰인의 되어보는 형태로 의뢰인의 생각을 이해할 수 있다. 사용자처럼 생각하기 위해서는 사용자와 함께 일할 필요가 있다.

4. 요구사항 문서화

최고의, 유일한 요구사항 문서는 작동하는 코드. 하지만, 문서로 정리하지 않아도 된다는 뜻은 아님

문서는 목표가 아니고, 의뢰인에게 승인해달라고 들이밀 수 있는 것도 아님. 문서는 안내역할을 하는 이정표

다만, 너무나 자세한 명세를 작성하지 않도록 함.



5. 용어사전 관리하기

모든 사람이 일관성을 위해 동일한 용어 사전을 사용해야 커뮤니케이션 비용이 줄어듦 온라인 문서에 사전 형태로 관리하는 것이 좋음

6. 불가능한 퍼즐 풀기

문제를 풀 때, 제약 조건인 틀을 이해하는 것이 가장 중요하다. 또한 퍼즐의 답은 주어진 자유도 안에서 발견된다.

7. 자신만의 방법에서 빠져나오기

- 1. 잠깐 딴짓을 해보라.
- 2. 이 질문에 답해보자.
 - a. 왜 이 문제를 풀고 있는가?
 - b. 문제를 풀어서 얻는 게 무엇인가?
 - c. 풀려고 하는 문제가 특수한 경우인가? 특수한 경우를 없앨 수는 없는가?
 - d. 관련 문제 중 내가 풀 수 있는 더 간단한 문제는 없는가?

8. 행복은 준비된 사람에게 찾아온다.

평소에 무의식 영역에 원료를 많이 주입해야 함. 어떤 원료가 부족한지는....

함께 일하기

1. 짝프로그래밍

짝프로그래밍/몹 프로그래밍을 하다 보면 한명은 큰 관점에서 문제를 바라보고 한 명은 작은 코드관점에서 문제를 바라볼 수 있어서 매우 효율적

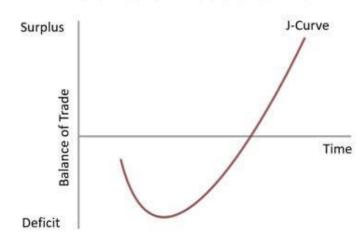
다만, 아래를 고려할 것

- 1. 코드를 짜는 거지 자아를 쌓는 게 아니다. 누가 가장 똑똑한 지 겨루는 것이 아니다. 모두 각자 뛰어난 부분이나 장단점이 있다.
- 2. 소규모로 시작할 것
- 3. 코드만 비판하고 사람을 비판하지 말 것. 틀렸어가 아니고, 여기를 한번 볼까요?
- 4. 자주 회고하고 개선할 점, 시도해 볼 것을 찾을 것
- 5. 다른 사람의 관점을 듣고 이해하려고 노력하라. 다른 것은 틀린 것이 아니다.

2. 애자일

애자일은 명사가 아님. 애자일은 방식에 관한 것 애자일 프로세스라는 것은 없음. 애자일은 어떤 것을 할지가 아니라, 어떤 것을 추구해야 할 지에 가까움

Balance of Trade J-Curve



- 1. 내가 어디에 있는 지 알아낼 것
- 2. 도달하고자 하는 곳을 향해 의미 있는 발걸음을 가능한 한 작게 옮길 것.
- 3. 어디에 도착했는지 평가하고, 망가뜨린 것이 있다면 고쳐라.

라이브로..

애자일

소프트웨어

수평문화