유효정_1장~2장

1장 도메인 주도 설계란 무엇인가

- 소프트웨어는 현실 세계의 프로세스를 자동화하거나 비즈니스 문제를 해결하기 위해 개발된다.
    - 이것을 소프트웨어의 도메인이라고 부른다.

- 도메인에 깊이 뿌리내리지 못한 소프트웨어는 향후의 변화에 제대로 대응할 수 없다.
    - 소프트웨어는 코드로 구성되어 있지만 너무 코드에 몰두해서 시간을 보내거나 단순한 오브젝트와 메서드 관점에서 보면 안된다.
    - 좋은 소프트웨어를 만들기 위해서는, 그 소프트웨어가 무엇에 관련된 것인지를 알아야 한다.
    - 소프트웨어의 전반적인 목적은 특정 도메인의 일이 더 잘 굴러가도록 하는 데 있고, 이는 그 소프트웨어가 대상 도메인과 조화를 이루어야 가능해진다.
    - 그렇지 않으면 도메인에 변형을 일으키고 오동작과 피해를 입히며 심지어는 혼란을 초래한다.

- 도메인이란 무엇인가
    - 도메인 모델은 특정한 다이어그램이 아니라, 그 다이어그램이 전달하고자 한 아이디어다 - 에릭 에반스
    - 도메인 전문가의 머리에 들어 있는 지식 중 선택적으로 추상화하여 엄격하게 조작화 된 것

- 모델이란 무엇인가
  - 대상 도메인에 대한 내부적 표현
  - 설계와 개발 프로세스 내내 모델은 반드시 필요하다.
  - 하나의 도메인에서 모델을 추출해 낼 때 필요없는 부분을 제외시켜야 하는데 이 부분이 상당히 도전적인 작업이다.
  - 정보를 조직화, 체계화하고 이것을 작은 부분으로 나누고 그 조각들을 다시 논리적 모듈로 그룹화한 다음 한번에 하나씩 선택해서 다루어야 한다.
  - 에를 들면, 은행 소프트웨어는 고객의 주소가 필요하지만고객의 눈 색깔은 필요 없다.
  - 도메인의 복잡성을 다루기 위해선 모델이 필요하다.
  - 우리는 모델을 사용해서 도메인 전문가, 설계자와 개발자들과 의사소통해야 한다

- 코드 설계 vs 소프트웨어 설계
  - 소프트웨어 설계는 집의 구조를 만드는 것 - 큰 그림
  - 코드 설계는 어떤 벽에 그림을 걸지 정하는 것 - 세부 사항
  - 소프트웨어 설계를 바꾸는 것은 코드 설계를 바꾸는 것보다 훨씬 큰 작업, 하지만 좋은 제품은 좋은 코드 설계 없이는 불가능하다.

- 소프트웨어 설계법
  - 폭포수 설계 방법
    - 업무 전문가 → 업무 분석가 → 개발자
    - 단점
      - 단방향으로 흘러가며, 피드백을 전달할 수 없다
      - 모든 요구사항을 프로젝트 초기에 확정하려고 시도하지만 프로젝트 초기에 완벽한 모델을 만들기는 힘들다.
      - 설계를 결정할 때 분석 마비 현상이 일어날 수 있다.
  - 익스트림 프로그래밍 (애자일 방법론)

- 미리 설계를 결정하지 않고, 구현을 매우 용이하게 변경할 수 있는 유연성을 확보한 상태에서, 비즈니스 이해관계자의 지속적인 참여와 수많은 리팩토링 작업을 통해 개발을 반복적으로 수행
- 단점
  - 애자일 방법론은 단순성을 추구하지만, 각 개인이 받아들인 단순성의 의미가 제각각
  - 견고한 설계 원칙 없이 지속적인 리팩토링을 거치게 되면 이해하기 어렵고 변경하기 어려운 코드가 양산 될 수 있음.

도메인 지식 쌓기

- 개발 프로세스가 도메인의 복잡한 문제를 모델링하고 구현할 수 있는 능력이 유지 가능한 방식으로 획기적으로 높아진다.
- 개발하는데 필요한 모델을 구축하려면 필수 정보를 골라내 일반화할 필요가 있다.
- 필요한 모델을 뽑아내는 과정
  - 비행기는 어떤 곳에서 이륙해서 다른 곳에서 착륙한다.
  - 각 비행기들은 비행 계획이 할당되어 있고, 각 비행기는 항로를 따른다는 정보를 얻게 되면, 더 이상 비행기를 출발점과 도착점과 연결 시킬 것이 아니라, 비행기를 항로와 연결시켜야 함. 그리고 항로는 출발점과 도착점이 있다는 것으로 생각하게 됨.
  - 그런데 실제로 항로란 출발에서 도착에 이르는 어떤 휘어진 선들이 여러개 모여 구성되어 있다는 것을 듣게 된다면. 항로에 하나의 출발점, 도착점이 있다고 받아들일게 아니라, 여러 고정지점 중 하나로 받아들이게 된다.
- 개발자와 도메인 전문가는 이야기를 통해 서로 지식을 교환해야 한다. 개발자는 올바르게 질문해야 하며 올바르게 정보를 가공하여 모델을 함께 만들어내야 한다.

2장 유비쿼터스 언어

소프트웨어 아키텍트, 개발자, 도메인 전문가로 구성된 설계팀은

1. 자신들의 행동을 통합하고
2. 모델 작성과, 작성된 모델의 코드화를 도와줄

언어가 필요하다.

공통 언어의 필요성

- 개발자와 도메인 전문가는 사용하는 언어가 달라서 의사소통에 어려움을 겪을 수 있다.

  [개발자] 클래스, 메서드, 알고리즘, 패턴, 클래스간 관계 (상속, 다형성)

  [도메인 전문가] 비행기, 경로, 고도, 위도, 경도

- 모델을 만들 때 아래와 같은 정보를 공유해야 한다.
  1. 모델에 대한 아이디어
  2. 모델에 포함되어야 할 요소
  3. 각 요소들을 어떻게 연결할 것인지
  4. 각 요소들 중 어떤 것들이 관계가 있고 어떤 것들이 관계가 없는지

- 모델을 이야기하고 정의할 때 같은 언어로 이야기 해야 한다 → 그럼 어떤 언어를 써야 하느냐?
- 도메인 주도 설계의 핵심 원칙은 `모델 기반의 언어를 사용하는 것`이다.

모델 기반의 언어

- 모델은 소프트웨어와 도메인이 서로 교차하는 지점이다
- 도메인 기반 모델을 표현한 언어가 프로젝트 전반에 걸쳐 모든 사용자에 의해 항상 사용 되어야 한다
  - 유비쿼터스 언어 (언제 어디서나 동시에 존재하는)
  - 모든 의사소통의 순간에서, 모델 기반의 언어를 사용하여 지식을 교환하고, 모델을 고안하고, 말하고 쓰고 다이어그램을 그려야 한다.
- 도메인과 설계를 정의할 핵심 개념을 찾아 거기에 해당하는 적절한 단어를 찾고 사용할 필요가 있다.

- 모델에 부합하도록 클래스, 메서드, 모듈의 이름을 변경하여 코드를 리팩터링 한다.
- 늘 쓰는 단어의 의미는 서로합의해서 혼란을 해소하라
- 도메인 전문가는 어색하거나 오해를 일으키는 용어 및 구조에 반대해야 한다
- 개발자는 설계에서 나타날 수 있는 모호함이나 불일치를 늘 검토해야 한다.

유비쿼터스 언어 만들기

- 유비쿼터스 언어를 사용하지 않으면
  - 장황하거나
  - 어떤 사실을 너무 돌려서 얘기하거나
  - 너무 상세하게 설명하려고 들거나
  - 잘못된 개념을 가지고 접근하기도 한다.
- 유비쿼터스 언어를 사용해야
  - 명확한 의사소통을 할 수 있다.
- 개발자는
  - 개발자만의 용어는 사용을 최대한 자제하고 유비쿼터스 언어를 사용해야 한다.
  - 모델의 주요 개념을 코드로 구현해 봄으로써 모델을 코드로, 언어를 코드로 매핑해보기를 권한다.