Domain-Driven Design

Tackling Complexity in the Heart of Software

복잡한 도메인(문제 영역)을 이해하고 모델링하고, 공통 언어로 소통하는데 초점을 맞추는 방법론.

- DDD는 모델을 동작하게 만들어 애플리케이션의 문제를 해결한다.[p.62]
  - 지식 탐구 → Ubiquitous Language 사용 → (단일) 모델 생성 → 모델과 구현을 밀접하게 연관시킴
- DDD의 목표는 **기술보다는 도메인에 대한 모델에 집중해 더 나은 소프트웨어를 만들어내는 것이다.**[p.154]

본문 내용 출처 : Eric Evans. 『도메인 주도 설계』. 이대엽(역). 위키북스, 2011.

Domain-Driven Design, Modeling, Design

서문

서문

1부 동작하는 도메인 모델 만들기

1부 동작하는 도메인 모델 만들기

2부 모델 주도 설계의 기본 요소

Part II: The Building Blocks of a Model-Driven Design



구현을 모델과의 밀접한 관계로 유지하기 위한 **모델링과 설계의 우수 실천 법**을 적용해야 한다. …

도메인 주도 설계 과정을 탄력성 있게 만들려면 개발자들은 **잘 알려진 근본 원리들**이 **어떻게 Model-Driven Design**을 뒷받침하는지 이해해야 한다. …

3개의 장은 **패턴 언어로**로 구성돼 있으며, 미묘한 모델의 특징과 설계 의사결정이 어떻게 도메인 주도 설계 과정에 영향을 주는지 보여주겠다 …

정교한 모델은 **가장 근본적인 사항에 관심을 가질 때만** 비로소 복잡성을 헤쳐나갈 수 있으며, 이는 팀에서 확신을 갖고 결합할 수 있는 상세 요소라는 결과로 나타난다.

Chapter

- 04 도메인의 격리

- 05 소프트웨어에서 표현되는 모델
- 06 도메인 객체의 생명 주기
- 07 언어의 사용: 확장 예제

3부 더 심층적인 통찰력을 향한 리팩터링

Part III: Refactoring Toward Deeper Insight

- 유용한 모델을 성공적으로 개발하기 위해 명심해야 할 세 가지 관건
  - 정교한 도메인 모델은 만들 수 있으며, 노력을 들일 만한 가치가 있다.
  - 해당 도메인을 학습하는 개발자와 도메인의 전문가의 **긴밀한 참여**와 **반복적인 리팩터링** 과정 없이 유용한 모델을 개발하기란 쉽지 않다.
  - 유용한 모델을 효과적으로 구현하고 사용하려면 **정교한 설계 기술**이 필요할지도 모른다.

리팩터링 수준

Levels of Refactoring



시스템의 생존력에 가장 큰 영향을 미치는 리팩터링은 (기술적인 관점보다) **도메인에 대한 새로운 통찰력을 얻었을때 수행**하거나 코드를 사용해서 **모델이 표현하고자 하는 바를 명확하게 드러내고자 수행**하는 경우다. (= 심층 모델을 향한 리팩터링)

리팩터링의 목표는 개발자가 단순히 코드가 수행하는 바를 이해하는 것뿐만 아니라 **왜 그렇게 수행되는지를 이해하고 도메인 전문가와의 의사소통에 이를 연관**시키는 것이다.

심층 모델

Deep Models

도메인의 **피상적[(본질적인 현상은 추구하지 아니하고 겉으로 드러나 보이는 현상에만 관계하는 것)]인 측면은 배제**하고 도메인 전문가의 **주요 관심사**와 **가장 적절한 지식**을 알기 쉽게 표현하는 모델이다. **이 정의가 추상화를 의미하는 것은 아니다**. 심층 모델이 일반적으로 추상적인 요소를 포함하기는 하지만 **문제의 핵심을 관통하는 구체적인 요소** 또한 포함할 수 있다.

도메인과 조화를 이루는 모델에서는 융통성, 단순함, 설명력을 얻을 수 있다. 그러한 모델이 공통적으로 지니고 있는 한 가지 특징은 **업무 전문가가 즐겨 쓰는 단순하지만 충분히 추상적인 언어**가 존재한다는 것이다.

심층 모델/유연한 설계

Deep Model/Supple Design

10 유연한 설계

Supple Design은 변경을 촉진할 뿐 아니라 **모델 자체의 개선**에도 기여한다. Model-Driven Design을 지탱하는 두 개의 축이 있다. 심층 모델은 [1]**설계의 표현력을 부여**한다. 그와 동시에 개발자가 여러 가지 시도를 할 수 있을 정도로 설계가 유연하고 개발자가 무슨 일이 일어나고 있는지 파악할 수 있을 만큼 설계가 명확하다면 [2]**모델의 발견 과정에 통찰력을 제공**할 수 있다.

발견 과정

The Discovery Process

- 09 암시적인 개념을 명확하게 만들기 : 도메인의 중심 개념을 찾아 설계에 반영하는 방법을 다룬다
- 10 유연한 설계 : 쉽게 확장하고 변경할 수 있는 소프트웨어를 작성하는 방법을 다룬다
- 11 분석 패턴의 적용, 12 모델에서 디자인 패턴 연결 : 모델링 삽질을 줄일 수 있는 방법 소개. 이러한 패턴은 바로 사용 가능한 해결책은 아니지만 면밀한 지식 검토 과정에 도움을 주고 조사해야 할 범위를 줄여준다.

Chapter

- 08 도약
- 09 암시적인 개념을 명확하게 만들기
- 10 유연한 설계
- 11 분석 패턴의 적용
- 12 모델에서 디자인 패턴 연결
- 13 더 심층적인 통찰력을 향한 리팩터링

4부 전략적 설계

Part IV: Strategic Design

Chapter

- 14 모델의 무결성 유지
- 15 디스틸레이션
- 16 대규모 구조
- 17 전략의 종합

From:
https://wiki.ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=domain-driven_design&rev=1708184847

Last update: **2026/10/24 07:03**