

03 모델과 구현의 연계

종이에 기록된 모델과 동작하는 소프트웨어 간의 괴리를 어떻게 좁힐것인가?

Domain Driven Design, Modeling, Design

Model-Driven Design

모델과 코드의 괴리를 좁히는데 여러 설계 방법론에서는 분석 모델의 필요성을 지지한다.

- 분석 모델 (Analysis model) : 소프트웨어 시스템에서 수행할 역할에 대해서는 전혀 고려하지 않은 채 **업무 도메인의 개념만**을 체계화하고자 해당 업무 도메인을 분석한 결과물
 - 설계와 뚜렷이 구분되고 다른 사람들이 만든다.
 - 오로지 **이해하기 위한 수단**으로만 간주되며, 구현 관심사와 섞일 경우 혼란만 초래하는 것으로 여겨진다.
 - 분석 모델은 설계상의 쟁점들을 염두에 두고 만들어진 것이 아니라서 모델과 설계의 연결을 분석 모델로 진행한다는 것은 매우 **비현실적**일 가능성이 높다.
- 순수하게 이론에만 치우친 분석 모델
 - **도메인의 이해**라는 가장 주된 목표에 미치지 못하기도 하는데, 중요한 발견은 언제나 설계/구현을 위해 노력하는 가운데 나타나기 때문이다. 매우 특이하고 미처 예상치 못한 문제는 늘 일어나기 때문이다.
 - 결과적으로 코딩이 시작되자마자 폐기되고 대부분의 문제를 다시 검토해야 한다.



설계 혹은 설계의 주된 부분이 도메인 모델과 대응하지 않는다면 그 모델은 그다지 가치가 없으며 소프트웨어의 **정확함**도 의심스러워진다. 동시에 모델과 설계 기능 사이의 복잡한 대응은 이해하기 힘들고, 실제로 설계가 변경되면 **유지보수**가 불가능해진다. 분석과 설계가 치명적으로 **동떨어지고**, 그에 따라 각자의 활동에서 얻는 **통찰력**이 서로에게 전해지지 않는다.

Model-Driven Design에서는 양쪽 모두의 목적을 달성하는 단일 모델을 찾기 위해 분석 모델과 설계를 나누는 **이분법은 채택하지 않는다**.

순수하게 기술적인 쟁점은 배제함으로써 설계상의 각 객체는 모델에서 기술한 **개념적 역할**을 수행한다. ...



(서로) 연계하는 과정에서 **기술적 고려사항** 탓에 분석이 심각하게 타협된 상태에 놓여서는 안 된다. 마찬가지로 **도메인 아이디어**는 반영하지만 소프트웨어 설계 원칙은 따르지 않은 서툰 설계를 받아 들여서도 안 된다. 이 접근법에는 분석과 설계 관점에서 모두 효과적인 모델이 필요하다.

모델이 **구현에 대해 비현실적**이거나 **도메인 핵심 개념을 충실하게 표현하지 않을 때** 새로운 모델을 찾아내야만 한다.

- How to?
 - 소프트웨어 시스템의 일부를 설계할 때는 **도메인 모델을 있는 그대로 반영해서** 설계와 모델의 대응을 분명하게 하라

- (모델) 지식탐구 → (코드) 리팩터링 → ... (반복)
 - **Ubiquitous Language**를 지원하는 것과 더불어 분석과 설계 두 가지 측면을 충분히 만족하는 **단 하나의 모델**을 만들어내야 한다.

모델링 패러다임과 도구 지원

Model-Driven Design이 성과를 내려면 인간의 오차 범위 내에서 정확하게 모델이 직접적으로 대응해야 하는데, 이를 만들어낼 수 있는 소프트웨어 도구가 뒷받침되는 모델링 패러다임 내에서 업무를 하는 것이 거의 필수적이다.

그 도구는 **OOP**. 객체 설계에서의 진정한 도약은 **코드가 모델의 개념을 표현**할 때 나온다.

반면, **Procedural** 언어에서는 프로그래머가 도메인의 개념을 생각할 수 있을지언정 프로그램 자체는 데이터를 기술적으로 조작하는 것에 지나지 않기에 의미를 많이 담지 못한다.

내부 드러내기: 왜 모델이 사용자에게 중요한가

Hands-on Modeler

실천적 모델러

From:
<http://wiki.ledyx.xyz/> - Ledyx WIKI

Permanent link:
http://wiki.ledyx.xyz/doku.php?id=domain_driven_design:03_binding_model_and_implementation&rev=1691495376

Last update: 2023/08/08 11:49

