

Anticorruption Layer

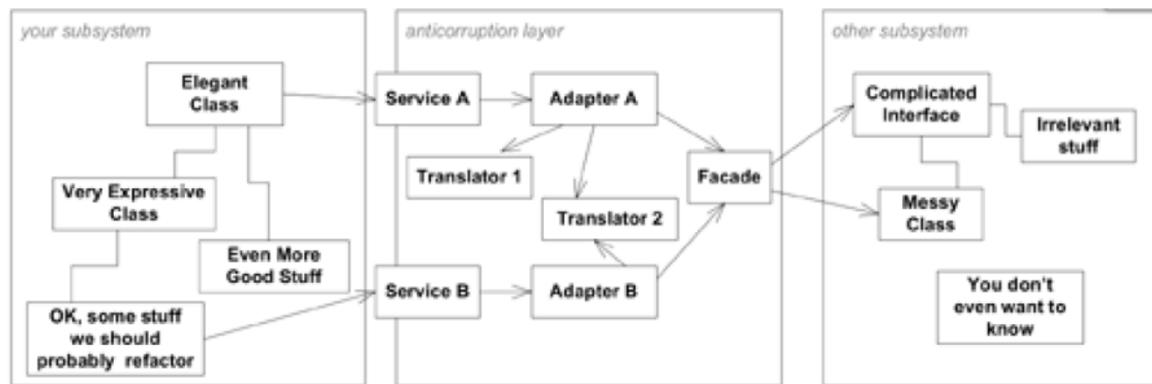
BoundedContext를 있는 수단

새로운 시스템을 개발하고, 이를 기존에 레거시 시스템과 결합할 경우 두 모델을 연계하는데 따르는 어려움 때문에 새로운 모델을 다른 레거시 시스템의 모델과 유사해지게끔 수정할 수 있고, 결과적으로 새로운 모델의 의도가 전체적으로 매몰돼 버릴 수 있습니다.

또, 시스템 끼리 연계할 때 시스템간의 미묘한 차이 때문에, 오류가 발생하거나 데이터베이스가 손상될 수 있습니다.

그러므로, 클라이언트 고유의 도메인 모델 측면에서 기능을 제공할 수 있는 격리 계층을 만들어야 합니다. 그리고 해당 계층에서는 필요에 따라 양방향으로 번역을 제공해야 합니다.

구현의 경우 어렵지만 FACADE, Adapter, Translator를 사용할 수 있습니다.



정확하게 뭔지 알수는 없지만, 예상을 해보자면

1 Facade를 통해서 하위 시스템에 대한 정보를 가져오고 (인터페이스 역할?)

→ 다른 시스템 모델에 맞춰서 작성

2 Adapter 및 Translator

→ 메세지 의미는 동일하지만, 정보를 새로운 시스템에 맞춰진 모델로 변환해주는 역할

3 Service A, Service B

→ 변환된 모델을 바탕으로 필요한 처리들 ?

주의할 사항

- 또 다른 시스템에서 **anticorruption layer**를 호출할 수 있게 바꿔야 할 수도 있음
- 통신 매커니즘은 실용적인 관점에서 접근하는 것이 좋음
- 자동화 테스트를 해두면 좋음
- 초기 비용이 많이 들지만, 그 다음부터는 유연하게 시스템을 운영할 수 있음

shared kernel, customer/supplier development team, conformist, antocorruption layer 같은 모든 통합에는 비용이 많이 들기 때문에

정말 통합해야 하는지 잘 확인해보아야 함.

Separate Ways

통합은 비용이 많이 들기 때문에, 관계가 필수적인 것이 아니라면 서로 관계를 끊을 수도 있습니다.

그래서 **Bounded Context**가 아무런 것과 관계를 맺지 않도록 선언하여 개발자들이 작은 범위 내에서 단순한 해결책을 찾을 수 있도록 유도하는 것도 방법입니다.

→ 요구사항을 철저히 파악해서 웬만하면 통합을 하지 않도록 유도,,,

Open Host Service

어떤 하위 시스템이 다른 여러 하위 시스템과 통합해야 할 경우 각 하위 시스템에 대한 번역기를 모두 조정해야 한다면, 팀 전체가 교착상태에 빠질 수 있습니다.

이를 해결하기 위해서, 프로토콜을 일련의 **Service**로 정의하고, 공개해서 모든 이들이 해당 프로토콜을 사용할 수 있도록 해야 합니다.

그리고 일반적인 요구사항들을 처리하며 확장해나가되 특정한 한 팀에서 요청해오는 독특한 요구사항들은 제외해야 합니다.

published language

두 도메인 모델간에 정보가 오갈 경우, 번역을 해야 하는데 이를 한 도메인에 맞추는 것은 좋지 않습니다. 두 도메인 모델 전부 각 문제 해결에 적합한 모델로 설계되었기 때문에 하나에만 맞추는건 좋지 않습니다.

Open Host Service에서는 여러 통합을 위해 표준화된 프로토콜을 사용합니다. 여기에 더 나아가서 그 언어를 공표함으로써, 서로 의사소통하고 해당 언어로 번역을 진행하면 좋습니다.

모델의 컨텍스트 전략 선택

팀에서 **Bounded Context**를 어디에 정의하고, 관계는 어떠할지 결정하고 팀원 모두가 이러한 사안을 알고 있어야 합니다.

경계 선택

Bounded Context의 경계를 설정하는 데에는 여러 요인들 사이에서 균형을 유지하는게 좋습니다.

규모가 큰 **Bounded Context**의 장점

- 단일화된 모델이 더 매끄러운게 선호될 경우
- 이해하기 쉬운 모델이 필요한 경우
- 두 모델간에 번역이 어려운 경우

규모가 작은 **Bounded Context**가 선호되는 경우

- 의사소통에 대한 부하가 줄어드는게 좋은 경우
- CI를 쉽게 구성하는 것이 좋은 경우
- 다양한 추상화 모델이 필요할 경우

이런 것들 사이에서 균형을 맞춰야 한다.

변경할 수 없다는 사실 인정하기

일부 레거시 시스템과, 일부 외부 시스템의 경우 명확하게 **Bounded Context**에 포함되지 않을것 같다면 빠르게 분리해야 합니다.

외부 시스템과의 관계

우선 **Separate ways**를 고려해보는게 좋습니다.

만약 정 통합이 필요하다면, **Conformist**나 **anticorruption layer** 가운데 하나를 선택할 수 있습니다.

근데 **conformist**가 되는건 시스템 개발시 선택사항이 제한되는걸 고려해야 합니다.

그리고, 번역이 얼마나 큰일이 될지도 파악해야 합니다.

타협점

단일화, 통합을 위해서 여러가지 전략이 존재합니다.

의사소통에 드는 노력과 통합이 주는 이점 사이에서 타협점을 찾아야 합니다.

팀의 의사소통 이행/능력, 시스템에 대해 통제력이 높을수록

- 단일 Bounded Context
- Shared Kernel
- Customer / Supplier Team
- Open Host Service

낮을 수록

- Anticorruption Layer
- Conformist
- Separate Ways

초기에 설정한 Bounded Context에 대한 결정을 변경할 수도 있습니다.

Separate Ways -> Shared Kernel

번역에 따르는 과부하가 너무 높고, 중복이 너무나도 명백하게 드러나면 Context를 병합해볼 수 있습니다.

궁극적으로 하나의 Context로 병합하는게 목표이더라도 Shared Kernel 부터 시작합니다.

자동화 테스트를 갖춘뒤에

코어 도메인부터 옮기지 말고, 중복되는 일부 작은 도메인부터 천천히 옮기면서 통합 및 번역을 검토하면 좋습니다.

아니면 새로운 공통 모델을 만들어 낼수도 있습니다.

그리고, 통합 과정에서 두 모델중 하나의 모델이 분명하게 선호된다면 선호하는 모델로 나아가는 것도 고려해볼만 합니다.

Shared Kernel -> Continuous Integration

Shared Kernel이 확장되는 중이라면 완전히 단일화도 고려해볼 수 있습니다.

코어를 병합할 때 선택할 수 있는 여지는 많지 않습니다.

한 모델을 고수하고 다른 모델을 변경하는 것, 하위 도메인을 대상으로 새로운 모델을 만드는 것 등을 고려할 수 있습니다.

Open Host Service -> Published Language

접근을 원하는 시스템이 늘어나면 유지보수 부담이 가중될 것입니다. 이 경우 Published Language를 갖춘 시스템 간의 관계를 공식화 해야 합니다.

- 표준 언어를 사용합니다.
- 표준 언어가 없다면 코어 도메인을 더 분명하게 만듭니다
- XML 같은 표준 교환 패러다임을 사용합니다.
- 언어, 아키텍처 등을 공표합니다.

디스틸레이션

규모가 큰 시스템의 경우 격리된 도메인조차도 관리할 수 없을 정도로 복잡해질 수 있습니다.

디스틸레이션은 훈합된 요소를 분리해서 본질을 유용한 형태로 뽑아내는 과정입니다.

분리된 요소들은 일련의 과정을 거쳐 Generic Subdomain과 Coherent Mechanism으로 더욱 값지게됩니다.

전략적 디스틸레이션에서는 아래와 같은 사항을 수행합니다.

1. 팀원들이 시스템 전체 설계와 해당 설계가 어떻게 조화될지 파악하도록 돕습니다.
2. Ubiquitous Language의 일부가 될 수 있게 핵심 모델을 식별해서 의사소통을 촉진합니다.
3. 리팩토링을 합니다.
4. 가장 중요한 모델 영역에 초점을 맞춥니다.

Core Domain (핵심 도메인)

규모가 큰 시스템에서는 구성요소가 무수히 많은데, 모두 복잡하고 필요한 것들이라 도메인의 본질적인 측면이 가려지거나 방치될 수 있습니다.

도메인 모델을 가치 있는 자산으로 만드려면 모델의 핵심적인 측면을 다루기 수월해야 하고 기능성을 만들어내는데 충분히 활용할 수 있어야 합니다.

도메인의 핵심부 설계의 경우 가장 중대한 사항으로 놓고 자원을 배분해야 합니다. 그리고 모든 사람들이 해당 사항을 식별하고 계획 및 개발이 이루어져야 합니다.

그러므로, 모델을 요약해야 합니다. 코어 도메인을 찾아 그것을 지원하는 다수의 모델과 코드로부터 쉽게 구별할 수 있는 수단을 제공해야 합니다.

코어 도메인에 가장 재능있는 인력을 할당하고 그에 따라 인력을 채용해야 합니다.