1. **Domain-Driven Design**

DDD được xây dựng xung quanh một khái niệm (concept) gọi là B**ounded Context**, tôi không biết dịch ra như thế nào có thể gọi đó là ranh giới giữa `modules` hoặc `services`trong hệ thống theo nghiệp vụ mà nó đại diện. Có nghĩa là mỗi bounded context đại diện cho một nghiệp vụ duy nhất và không thể thay thế trong hệ thống

Nó tách hệ thống thành các domain riêng và triển khai chúng độc lập

Các bước để triển khai DDD :

* Domain spliting : chia, tách yêu cầu, làm rõ các yêu cầu về hệ thống
* 1 yêu cầu về hệ thống có thể chia thành các phần

Actor : người thực hiện

Command : yêu cầu, mệnh lệnh đối với hệ thống

Event : những gì sẽ được xảy ra, được thực hiện bởi hệ thống

Event

Tạo đơn hàng và theo dõi

Command

Nhận và vận chuyển đơn hàng

Actor

Bưu tá

* Domain modeling : thiết kế các domain
* Từ nhiều những yêu cầu về hệ thống ta tách những actor, command, event và nhóm những phần liên quan thành những domain riêng biệt

Đối với VTP thì :

Main domain

Support domain

(cùng level với core domain)

SMS domain

External domain

TMS

OMS

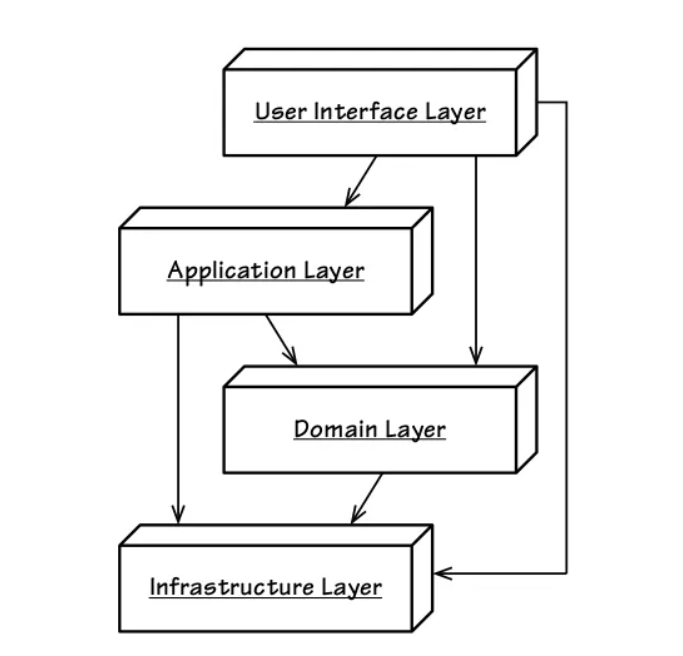
Core domain

Ecormerce domain

Interact

1. **Kiến trúc phân lớp trong DDD (4 lớp)**

khi các đoạn code liên quan đến nghiệp vụ được trộn lẫn giữa các tầng lại với nhau, nó trở nên vô cùng khó khăn cho việc đọc cũng như suy nghĩ về chúng. Các thay đổi ở giao diện người dùng cũng có thể thực sự thay đổi cả logic nghiệp vụ. Để thay đổi logic nghiệp vụ có thể yêu cầu tới truy vết tỉ mỉ các đoạn mã của giao diện người dùng, CSDL, hoặc các thành phần khác của chương trình. Mô hình phát triển hướng đối tượng trở nên phi thực tế. Do đó, hãy phân chia một chương trình phức tạp thành các LỚP. Phát triển một thiết kế cho mỗi LỚP để chúng trở nên gắn kết và chỉ phụ thuộc vào các tầng bên dưới. Dưới đây là giải pháp kiến trúc chung cho DDD.

* User Interface: Chịu trách nhiệm trình bày thông tin tới người sử dụng và thông dịch lệnh của người dùng. Có thể hiểu là các sự kiện xảy ra trên giao diện khi được trigger sẽ được dịch thành lệnh và xử lý ở các tầng dưới.
* Applicatioin Layer: Tầng này được thiết kế khá mỏng (ít xử lý logic) phối hợp các hoạt động của ứng dụng. Nó không chứa logic nghiệp vụ. Nó không lưu giữ trạng thái của các đối tượng nghiệp vụ nhưng nó có thể giữ trạng thái một tiến trình của ứng dụng. Chúng ta có thể hình dung phần này gần giống với các Controller trong mô hình MVC chỉ làm nhiệm vụ chuyển tiếp các task đến nơi cần xử lý.
* Domain Layer: Tầng này chứa thông tin về các lĩnh vực. Đây chính là trái tim của phần mềm. Trạng thái của đối tượng nghiệp vụ được giữ tại đây.
* Infrastructure Layer: Tầng này đóng vai trò như một thư viện hỗ trợ cho tất cả các tầng còn lại. Nó cung cấp thông tin liên lạc giữa các lớp, cung cấp chức năng lưu trữ các đối tượng nghiệp vụ, chứa các thư viện hỗ trợ cho tầng giao diện người dùng...

1. Mô hình triển khai mẫu cho TMS

Mô hình triển khai quy trình thực hiện 1 order creation từ phía OMS và tracking từ phía người dùng

