Paweł Rajba
pawel@ii.uni.wroc.pl
http://www.kursy24.eu/

Architektura aplikacji

Agenda

- Wprowadzenie
- Architectural styles
- Domain Driven Design
- Patterns of Enterprise Application Architecture

Wprowadzenie

- O architekturze systemu można mówić na różnym poziomie szczegółowości,
- … z różnych punktów widzenia
- Interakcja z danymi jest jednym z aspektów
  - dane przepływają przez wszystkie warstwy systemu – od bazy danych do interfejsu użytkownika

Architectural styles

- Bardzo ogólne podejście do architektury
- Opisują architekturę z różnych punktów widzenia
- Podział na kategorie
  - Communication
    - Service-Oriented Architecture (SOA), Message Bus
  - Deployment
    - Client/Server, N-Tier, 3-Tier
  - Domain
    - Domain Driven Design
  - Structure
    - Component-Based, Object-Oriented, Layered Architecture

Architektura klient-serwer

- Dwie strony komunikacji
  - Klient – strona żądająca dostępu do danych, usługi
    - aktywny: wysyła żądanie do serwera i oczekuje odpowiedzi
  - Serwer – udostępnia dane i usługi
    - pasywny: czeka na żądania od klientów
    - w momencie otrzymania żądania, przetwarza je, a następnie wysyła odpowiedź
- Rodzaje architektury klient serwer
  - dwuwarstwowa, trójwarstwowa, wielowarstwowa

Architektura dwuwarstwowa

- Przetwarzanie i składowanie danych odbywa się w jednym module
- Dawniej bardzo popularny, obecnie wypierany przez model trójwarstwowy
- W tym modelu jest zapewniony wielodostęp
- Tańsze niż w modelu wielowarstwowym, ale
  - Problem ze skalowalnością
  - Gdzie logika biznesowa?
  - Trudniej zapewnić różne sposoby prezentacji
    - np. web client, mobile client

Architektura trójwarstwowa

- Przetwarzanie i składowanie danych następuje w dwóch osobnych modułach
- Występują następujące warstwy
  - Prezentacji
  - Logiki biznesowej
  - Danych
- Skalowalne, łatwiej zapewnić różne formy prezentacji

Architektura wielowarstwowa

- Przetwarzanie, składowanie i inne operacje na danych odbywają się w wielu osobnych modułach
- Zwykle tworzy się hierarchie warstw, gdzie warstwa n komunikuje się z warstwami n-1 i n+1
- Ważne cechy podejścia warstwowego
  - Abstrakcja, enkapsulacja, podział funkcjonalności pomiędzy warstwy, reużywalność, luźne powiązania
- Główne zalety takiego podejścia
  - Abstrakcja, izolacja, zarządzalność, wydajność, reużywalność, testowalność

Architektura wielowarstwowa

- Przykładowy podział na warstwy
  - Warstwa interfejsu użytkownika
    - np. web client, mobile client
  - Warstwa zarządzania treścią, tzw. backend
    - np. CMS
  - Warstwa usług: web services & data services
    - np. dostarczanie danych innym modułom
  - Warstwa uwierzytelnienia i autoryzacji
    - np. delegacja na zupełnie zewnętrzny serwer
  - Warstwa pamięci masowej, czyli baza danych

Przetwarzanie danych

- W architekturze wielowarstwowej przetwarzanie może być
  - Na serwerze bazy danych (funkcje i procedury)
  - Na serwerze aplikacji
  - Rola hurtowni danych
- Zalety i wady poszczególnych rozwiązań

Patterns of Enterprise Application Architecture

- Sporo różnych wzorców, część dotyczy danych
- Katalog wzorców wg. guru Martina Fowlera
  http://martinfowler.com/eaaCatalog/
- Przyjrzymy się chwilę dwóm kategoriom
  - Domain Logic Patterns
  - Data Source Architectural Patterns
    - Pozostałe kategorie dot. będą omawiane przy okazji tematów, których dotyczą, np. przy omawianiu ORM

Patterns of Enterprise Application Architecture

- Domain Logic Patterns
  - Transaction Script
    - *Organizes business logic by procedures where each procedure handles a single request from the presentation.*
  - Domain Model
    - *An object model of the domain that incorporates both behavior and data.*
  - Table Module
    - *A single instance that handles the business logic for all rows in a database table or view.*
  - Service Layer
    - *Defines an application's boundary with a layer of services that establishes a set of available operations and coordinates the application's response in each operation.*

Patterns of Enterprise Application Architecture

- Data Source Architectural Patterns
  - Table Data Gateway
    - *An object that acts as a Gateway (466) to a database table. One instance handles all the rows in the table.*
  - Row Data Gateway
    - *An object that acts as a Gateway (466) to a single record in a data source. There is one instance per row.*
  - Active Record
    - *An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.*
  - Data Mapper
    - *A layer of Mappers (473) that moves data between objects and a database while keeping them independent of each other and the mapper itself.*

Literatura

- Software Architecture and Design
  http://msdn.microsoft.com/en-us/library/ee658093.aspx

Design principles

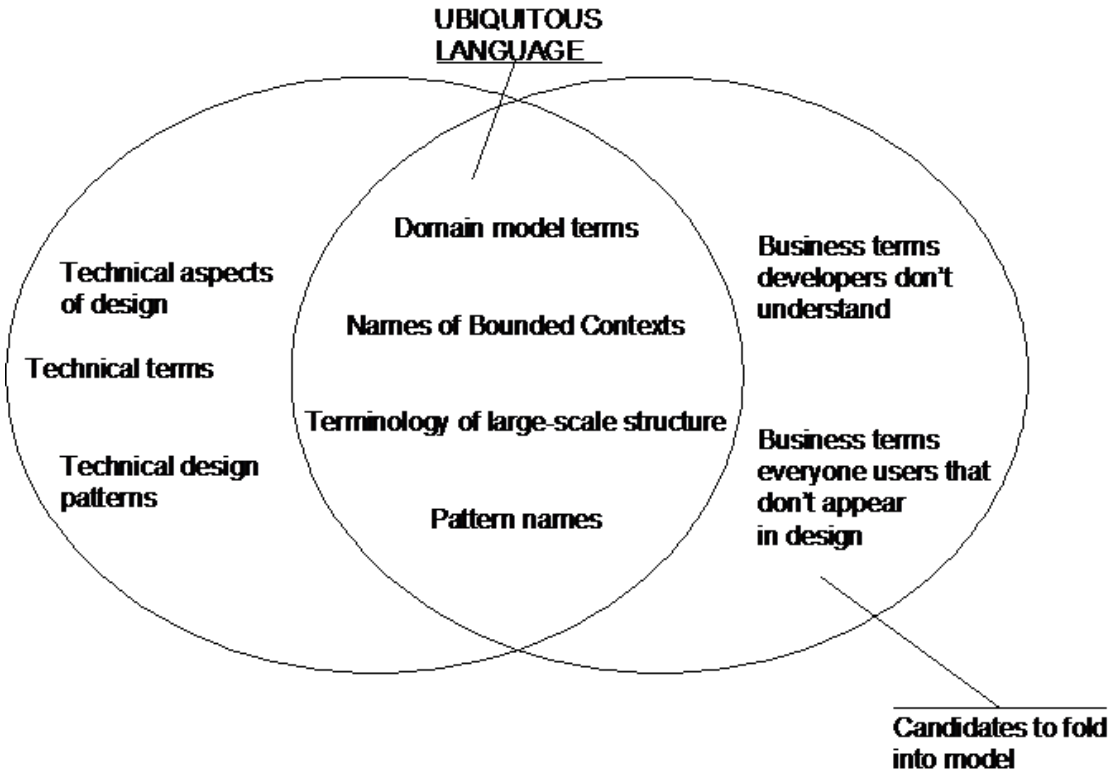
- SOLID
  - http://www.oodesign.com/design-principles.html
  - http://joelabrahamsson.com/a-simple-example-of-the-openclosed-principle/
- KISS
  - http://en.wikipedia.org/wiki/KISS_principle
- DRY
  - http://en.wikipedia.org/wiki/Don%27t_repeat_yourself
- SoC
  - http://en.wikipedia.org/wiki/Separation_of_concerns
- YAGNI
  - http://en.wikipedia.org/wiki/You_aren%27t_gonna_need_it
- Law of Demeter
  - http://pl.wikipedia.org/wiki/Prawo_Demeter

Domain Driven Design

- Podstawy DDD
  - Najważniejsze w DDD to dobre zrozumienie dziedziny, utworzenie modelu, czyli przybliżenia fragmentu rzeczywistości
    - User stories: as a [role], I want [feature] so that [benefit]
    - Use cases
  - Model to całość zagadnienia, na który składają się diagramy, przypadki użycia, obrazki ale również słowne opisy różnych powiązań, zależności i ogólnie wiedzy związanej z daną dziedziną

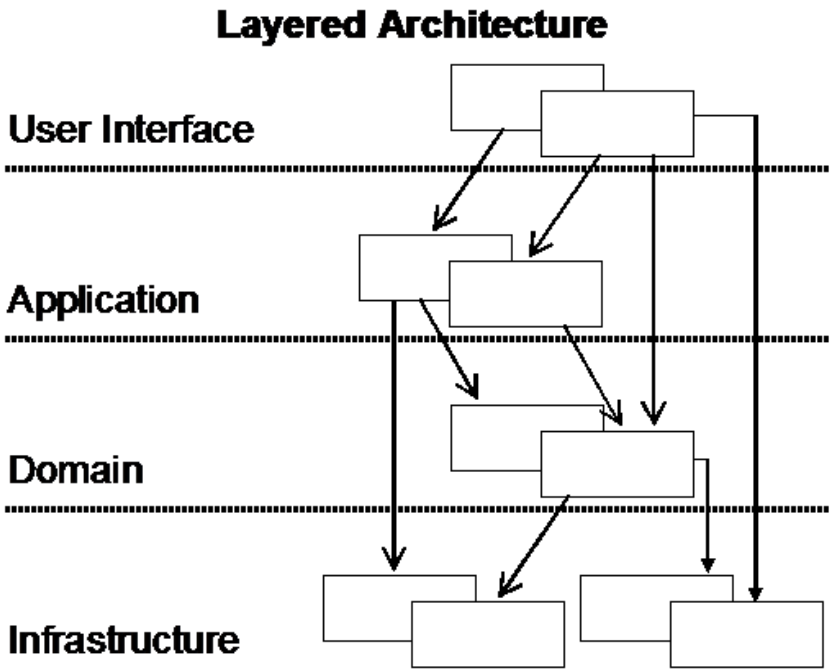
Domain Driven Design

- Podstawy DDD
  - Ubiquitous Language

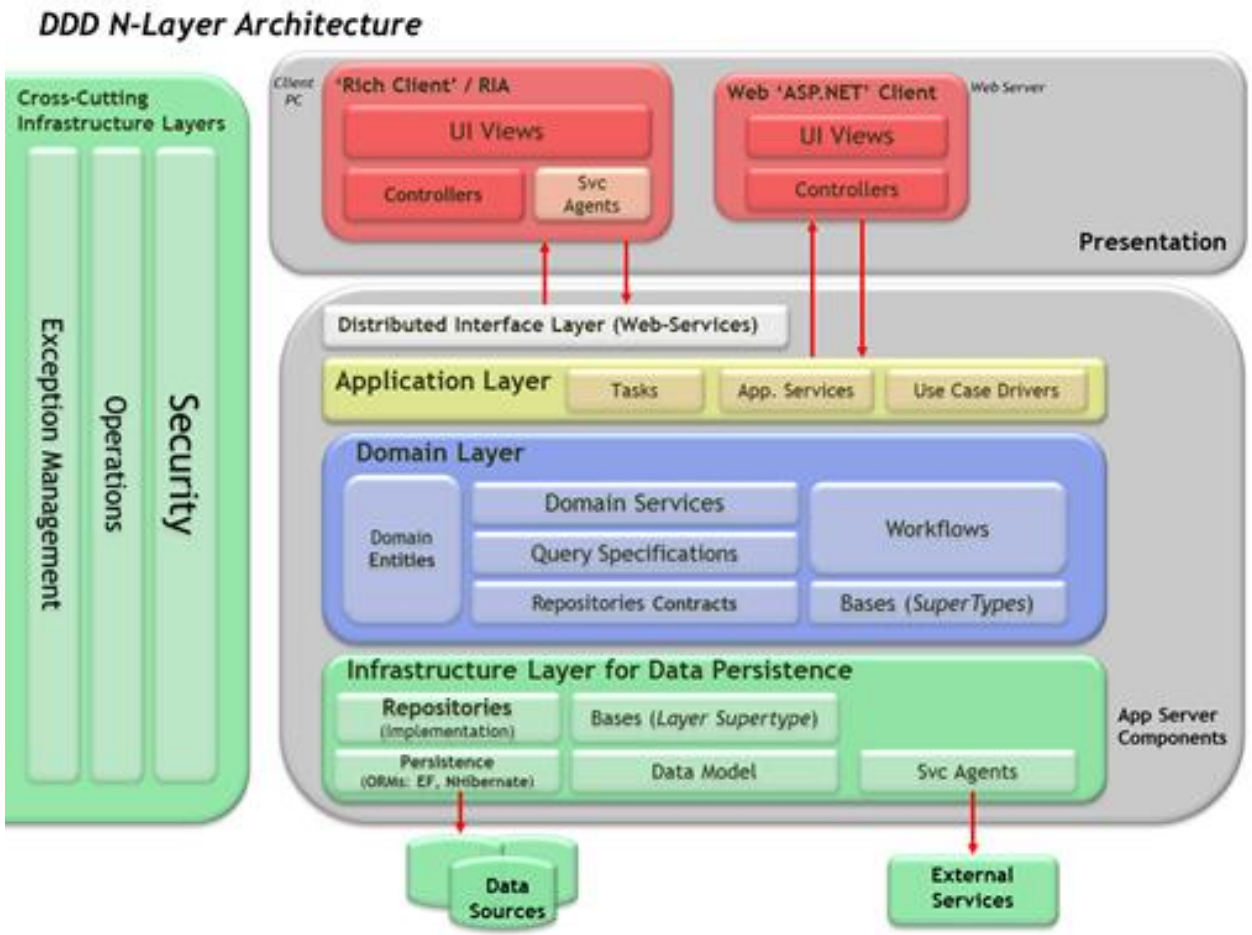


Domain Driven Design

- Architektura warstwowa
  - Presentation Layer (User Interface)
  - Application Layer
  - Domain Layer
  - Infrastructure Layer



Domain Driven Design



Źródło: http://blogs.msdn.com/b/cesardelatorre/archive/2010/03/26/our-brand-new-ddd-n-layer-net-4-0-architecture-guide-book-and-sample-app-in-codeplex.aspx

Domain Driven Design

- Podstawy DDD
  - Bounded Context
    - podział systemu na spójne obszary
    - znalezienie powiązań między nimi
    - każdy context ma swój Ubiquitous Language
    - Przykład:
      - Tworzymy system, który korzysta z danych systemu kadrowego
      - System dla kadr tworzy niezależny od naszego „bounded context”

Domain Driven Design

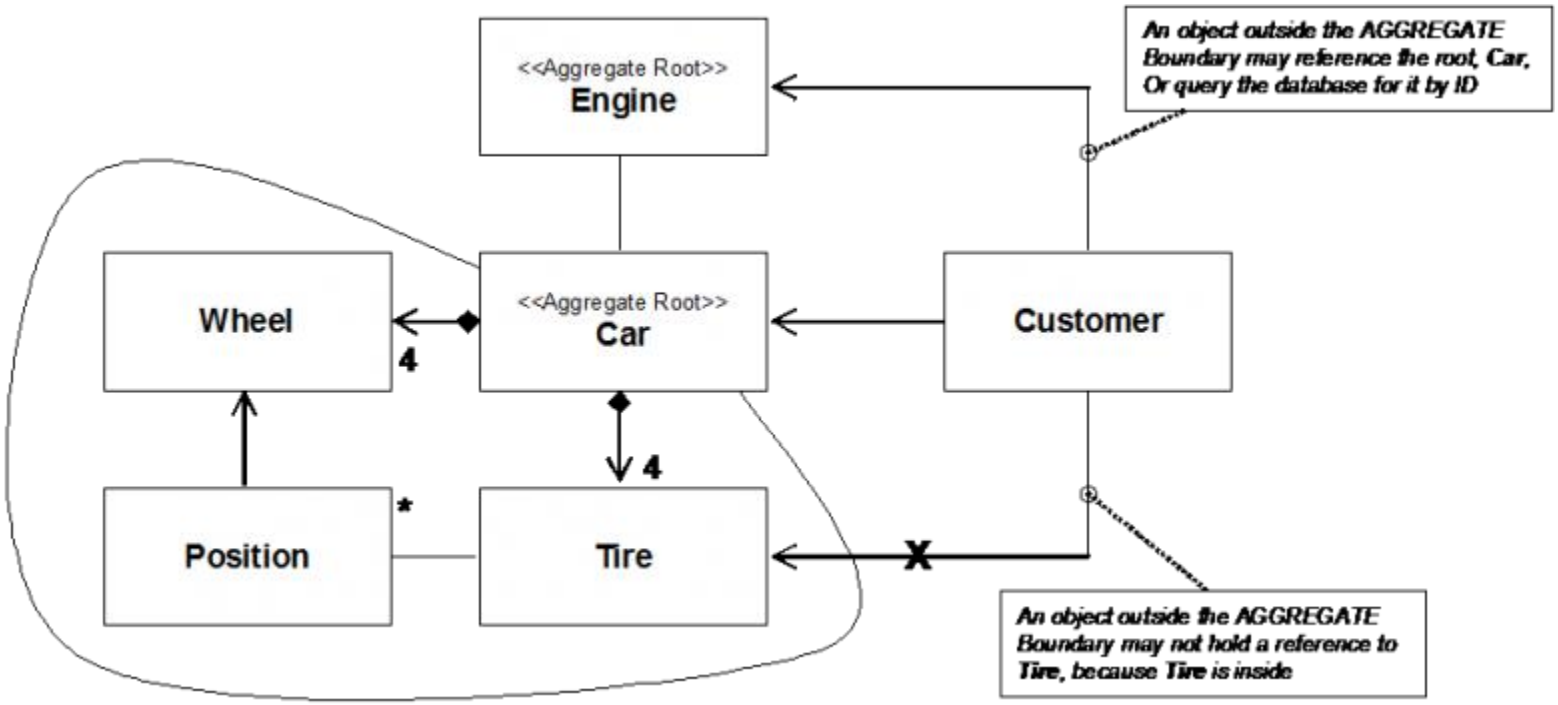
- Składowe modelu
  - Entities
    - pojedyncze „rzeczy”
    - mają identyfikator biznesowy i cykl życia
    - zawierają logikę operującą na encji
    - wartości atrybutów mogą się zmieniać
  - Value objects
    - nie mają ID ani cyklu życia
    - powiązane z encjami, opisują je
    - wartości atrybutów nie mogą się zmieniać

Domain Driven Design

- Składowe modelu, c.d.
  - Aggregates
    - Zbiór entities i value objects
    - Rozpatrywana jako „single business unit”
    - Dostęp tylko przez „aggregate root” (który jest entity)
      - Ale w wewnątrz mogą być referencje do innych agregatów
    - Operacje
      - Zapytania: tylko w kontekście „aggregate root”
      - Update: spójność w zakresie jednego agregatu
      - Delete: usunięcie aggregate root implikuje usunięcie całego agregatu

Domain Driven Design

- Składowe modelu, c.d.
  - Aggregates, przykład

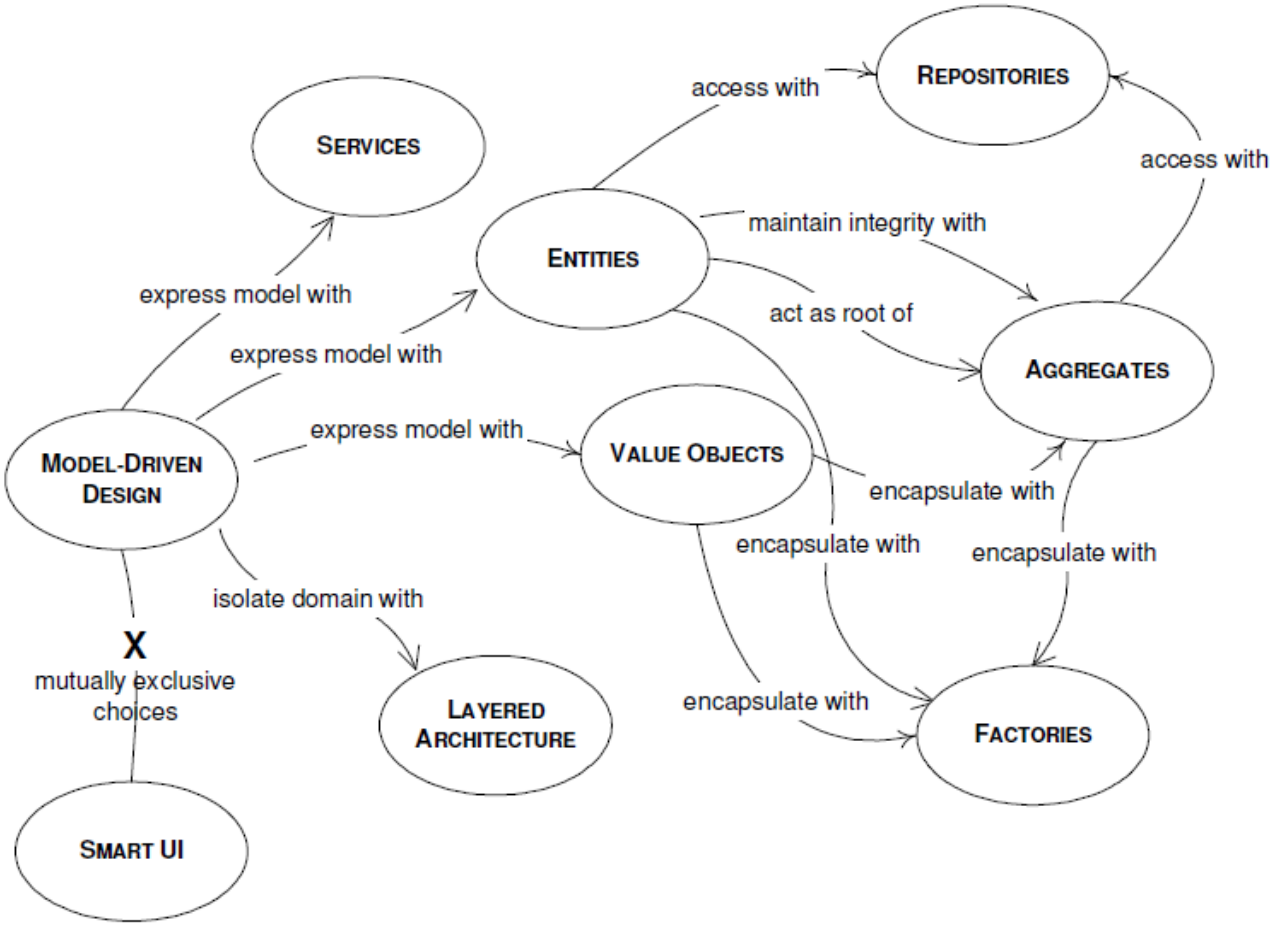


Domain Driven Design

- Charakterystyka, składowe
  - Domain Services
    - nie mają stanu, id, cyklu życia
    - mogą operować na więcej niż jednej encji
    - udostępniają operacje dostępne dla zewnętrznych systemów
  - Factories
    - tworzą encje i agregaty
  - Repositories
    - odpowiedzialne za utrwalanie encji i agregatów
    - wprowadzają separację modelu od utrwalania
  - Modules
    - Większe, spójne obszary z własnym interfejsem

Domain Driven Design

- Podsumowanie



Domain Driven Design

- Dodatkowe elementy
  - Dbałość o spójność modelu
  - Refaktoryzacja
  - Testowalność, utrzymywanie testów
    - jednostkowe
    - integracyjne
  - Continuos integration

Domain Driven Design

- Literatura
  - Wprowadzenie w temat
    http://msdn.microsoft.com/en-us/magazine/dd419654.aspx
    http://www.codeproject.com/Articles/56767/Domain-Driven-Design
    http://www.infoq.com/articles/ddd-in-practice
    http://leanagilechange.com/leanagilewiki/index.php?title=Domain_Driven_Design
  - Przegląd DDD
    http://www.infoq.com/minibooks/domain-driven-design-quickly