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

Architektura aplikacji

Agenda

- Architektura
- Architectural styles
- Patterns of Enterprise Application Architecture
- Design Principles
- SOLID

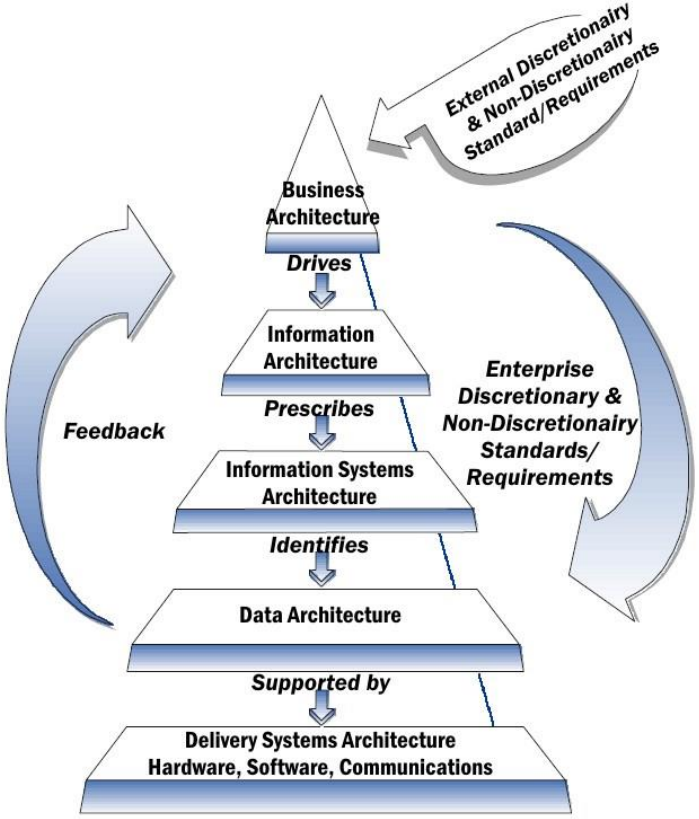
Architektura

Bass, Clements i Kazman, 2003:
„Architektura oprogramowania programu lub systemu informatycznego jest strukturą lub zbiorem struktur tego systemu, obejmującym elementy oprogramowania, widoczne dla pozostałych elementów oprogramowania oraz zależności między nimi.”

ISO/IEC 42010:2007:
„Architektura systemu informatycznego jest to podstawowa organizacja systemu wraz z jego komponentami, wzajemnymi powiązaniami, środowiskiem pracy i regułami ustanawiającymi sposób jej budowy i rozwoju”

Architektura

- O architekturze systemu można mówić na różnym poziomie szczegółowości i różnych punktów widzenia
  - Enterprise
  - Information
  - Integration
  - Infrastructure
  - Software
  - Security
  - …



https://en.wikipedia.org/wiki/NIST_Enterprise_Architecture_Model

Architektura

- Interakcja z danymi jest jednym z aspektów
  - Dane przepływają przez wszystkie warstwy systemu
    - Od bazy danych do interfejsu użytkownika
      - (lub odwrotnie)
    - Pomiędzy systemami
  - Przepływy jest automatyczne lub manualne
    - Dane wpisywane do Excela i wysyłane e-mailem

Architektura

- Dlaczego?
  - Łatwiej zrozumieć całość rozwiązania
    - W tym ułatwia zrozumienie, jak potencjalne zmiany wpłyną na rozwiązanie
  - Ułatwia tworzenie komponentów wielokrotnego zastosowania
  - Redukuje koszt utrzymania i rozwoju systemu
  - Pozwala kontrolować wymagania niefunkcjonalne
  - Jest niezbędna, aby zapewnić wysoką jakość oprogramowania

Architektura

- Dlaczego?

- Accessibility
- Adaptability
- Auditability and control
- Availability (see service level agreement)
- Backup
- Capacity, current and forecast
- Certification
- Compliance
- Configuration management
- Cost, initial and Life-cycle cost
- Data integrity
- Data retention
- Dependency on other parties
- Deployment
- Development environment
- Disaster recovery
- Documentation
- Durability
- Efficiency (resource consumption for given load)
- Effectiveness (resulting performance in relation to effort)
- Emotional factors (like fun or absorbing or has "Wow! Factor")
- Environmental protection
- Escrow
- Exploitability
- Extensibility (adding features, and carry-forward of customizations at next major version upgrade)
- Failure management
- Fault tolerance (e.g. Operational System Monitoring, Measuring, and Management)
- Legal and licensing issues or patent-infringement-avoidability
- Interoperability
- Maintainability (e.g. Mean Time To Repair - MTTR)
- Management
- Modifiability
- Network topology
- Open source
- Operability
- Performance / response time (performance engineering)
- Platform compatibility
- Privacy (compliance to privacy laws)
- Portability
- Quality (e.g. faults discovered, faults delivered, fault removal efficacy)
- Readability
- Reliability (e.g. Mean Time Between/To Failures - MTBF/MTTF)
- Reporting
- Resilience
- Resource constraints (processor speed, memory, disk space, network bandwidth, etc.)
- Response time
- Reusability
- Robustness
- Safety or Factor of safety
- Scalability (horizontal, vertical)
- Security (cyber and physical)
- Software, tools, standards etc. Compatibility
- Stability
- Supportability
- Testability
- Throughput
- Transparency
- Usability (Human Factors) by target user community
- Integrability ability to integrate components

https://en.wikipedia.org/wiki/Non-functional_requirement

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

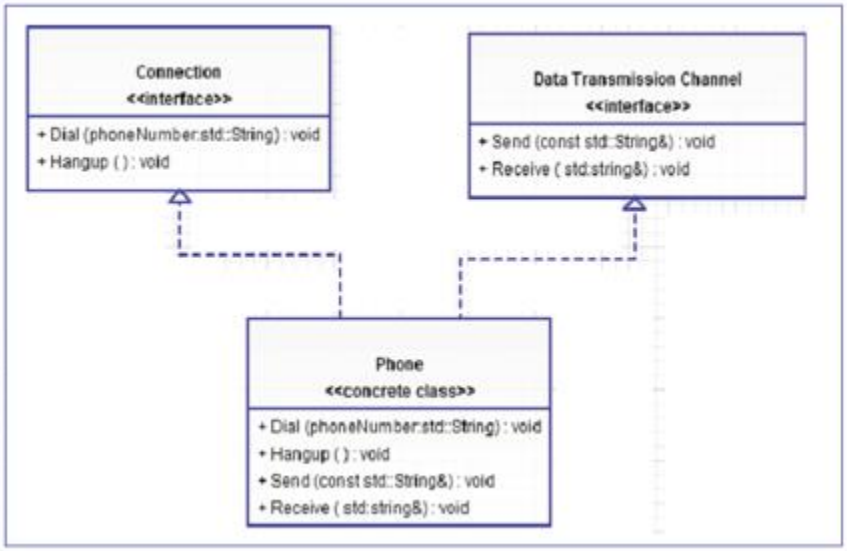
- Data Source Architectural Patterns
  - Table Data Gateway
    - *An object that acts as a Gateway (466) to a database table. One instance handles all the rows in the table.*
  - Row Data Gateway
    - *An object that acts as a Gateway (466) to a single record in a data source. There is one instance 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.*

Design principles: SOLID

- S: Single Responsibility Principle

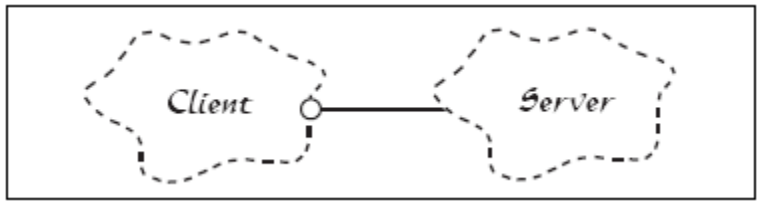
SRP says

"Every software module should have only one reason to change".

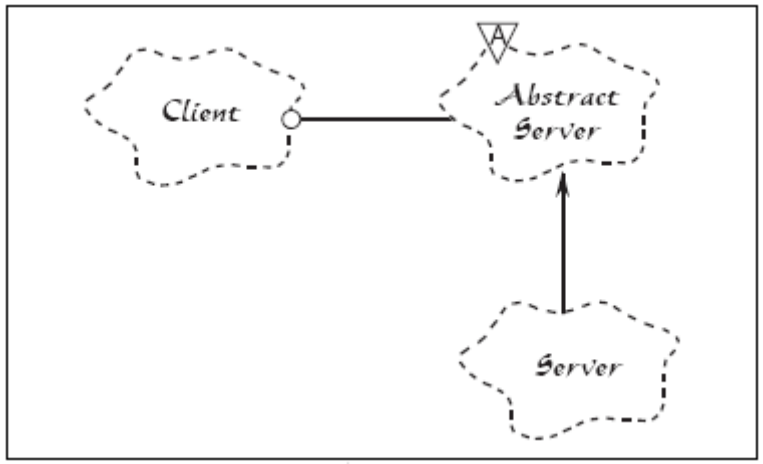


Design principles: SOLID

- O: Open/Closed Principle
  - Tu klienta trzeba zmienić

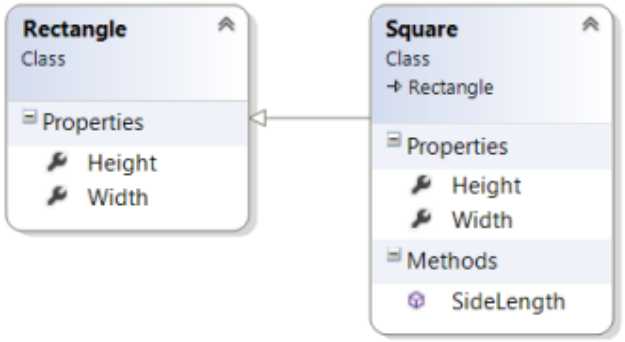


  - Tutaj można podmienić serwer bez zmiany klienta



Design principles: SOLID

- L: Liskov substitution principle




```
class Rectangle
{
    public virtual int Width { get; set; }
    public virtual int Height{ get; set; }
}
```

```
class Square : Rectangle
{
    public override int Height
    {
        get { return base.Height; }
        set { SideLength(value); }
    }

    public override int Width
    {
        get { return base.Width; }
        set { SideLength(value); }
    }

    // Both sides of a square are equal.
    public void SideLength(int value)
    {
        base.Height = value;
        base.Width = value;
    }
}
```

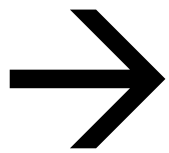
- Przekazujemy Rectangle jako parametr
- Liczymy pole dla Rectangle

https://mkatkoot.wordpress.com/2013/05/27/solid-liskov-substitution-principle/

Design principles: SOLID

- I: Interface Segregation Principle

```
interface IWorker
{
    void Work();
    void Eat();
}
class Worker: IWorker
{
    public void Work()
    {
    }
    public void Eat()
    {
    }
}
class Robot: IWorker
{
    public void Work()
    {
    }
    public void Eat()
    {
      throw new NotImplementedException();
    }
}
```

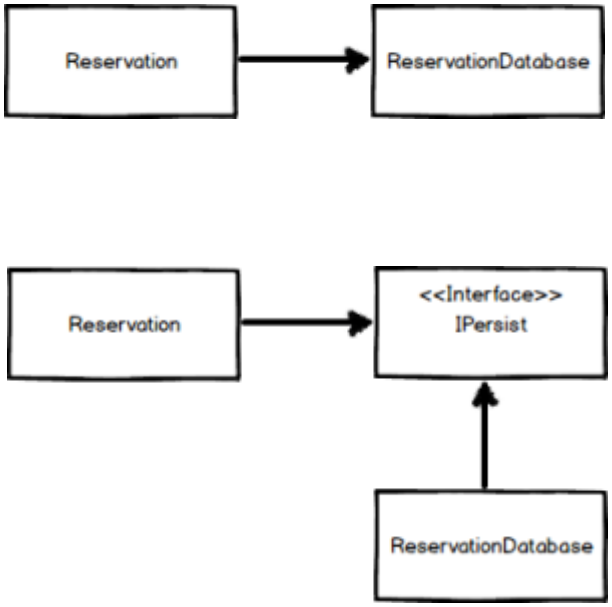


```
interface IWorkable
{
    public void Work();
}
interface IFeedable
{
    public void Eat();
}
class Worker: IWorkable, IFeedable
{
    public void Work()
    {
    }
    public void Eat()
    {
    }
}
class Robot: IWorkable
{
    public void Work()
    {
    }
}
```

http://www.pzielinski.com/?p=424

Design principles: SOLID

- D: Dependency Inversion Principle



- Other concepts:
  - Dependency Injection
  - Inversion of Control

http://ricardodsanchez.com/tag/solid-principles/

Design principles summary

- SOLID
  - http://www.oodesign.com/design-principles.html
  - http://joelabrahamsson.com/a-simple-example-of-the-openclosed-principle/
  - http://www.codeproject.com/Articles/567768/Object-Oriented-Design-Principles
- 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

Literatura

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