GUI - projektowanie interfejsów

dr Przemysław Juszczuk

Katedra Inżynierii Wiedzy, Uniwersytet Ekonomiczny w Katowicach

Wykład 5

Wzorce projektowe

Idea wzorców projektowych wywodzi się jeszcze z wczesnych lat osiemdziesiątych ubiegłego wieku, gdzie podstawowym językiem programowania obiektowego był Smalltalk. Samo pojęcie wzorca pojawiło się jeszcze wcześniej - w roku 1977, gdzie Christopher Alexander stosował ten termin w odniesieniu do budynków i miast.

Wzorce projektowe - definicja

Wzorce projektowe (ang. Design patterns) powiązane są ściśle z programowaniem obiektowym. - wielokrotne stosowanie obiektu w różnych projektach oraz pomiędzy wieloma programistami.

```
class Test {

   private:
      int liczba;
      String napis;

   public:
      bool setLiczba (int newLiczba) {
         // validation
         if (validation) {
            this->napis = newNapis;
            return true;
         }
      }

      int getLiczba () {
         return this->liczba;
      }

      bool setNapis (String newNapis) {
         // validation
         if (validation) {
            this->napis = newNapis;
            return true;
         }
      }

      String getNapis () {
         this->napis;
      }
}
```

Rysunek: Krótkie przypomnienie

Szablon wzorca

- nazwa wzorca - dobrana jest w taki sposób, aby szybko nasuwać skojarzenie z jego przeznaczeniem. Wszystkie stosowane nazwy powinny być anglojęzyczne. Oczywiście niektóre nazwy mogą być przetłumaczone na język polski bez powodowania niejednoznaczności.
- cel (określane także jako opis problemu) - pozwala wskazać sytuację, w jakiej należy stosować dany wzorzec projektowy. Czasami podawana jest także lista warunków, które powinny zostać spełnione, aby zastosowanie danego wzorca było możliwe.
- klasyfikacja i aliasy - aliasy to alternatywne nazwy wzorca. Natomiast klasyfikacja to nazwa jednej z dużych grup, do której należy dana struktura.

Szablon wzorca II

- motywacja - przykładowy scenariusz opisujący zaistniały problem oraz sposób jego rozwiązania przy zastosowaniu zaproponowanego wzorca.
- opis struktury wzorca - jeden z najistotniejszych elementów przedstawiający jego graficzną reprezentację. Chodzi tutaj przede wszystkim o zakres powiązań pomiędzy danym wzorcem a klasami w postaci diagramu klas UML.
- konsekwencje - efekty oraz koszty zastosowania wzorca w danym projekcie.
- przykład - przykład kodu, w którym zastosowany został wybrany wzorzec.

Wzorce kreacyjne

- Abstract factory;
- Builder;
- Prototype;
- Singleton;

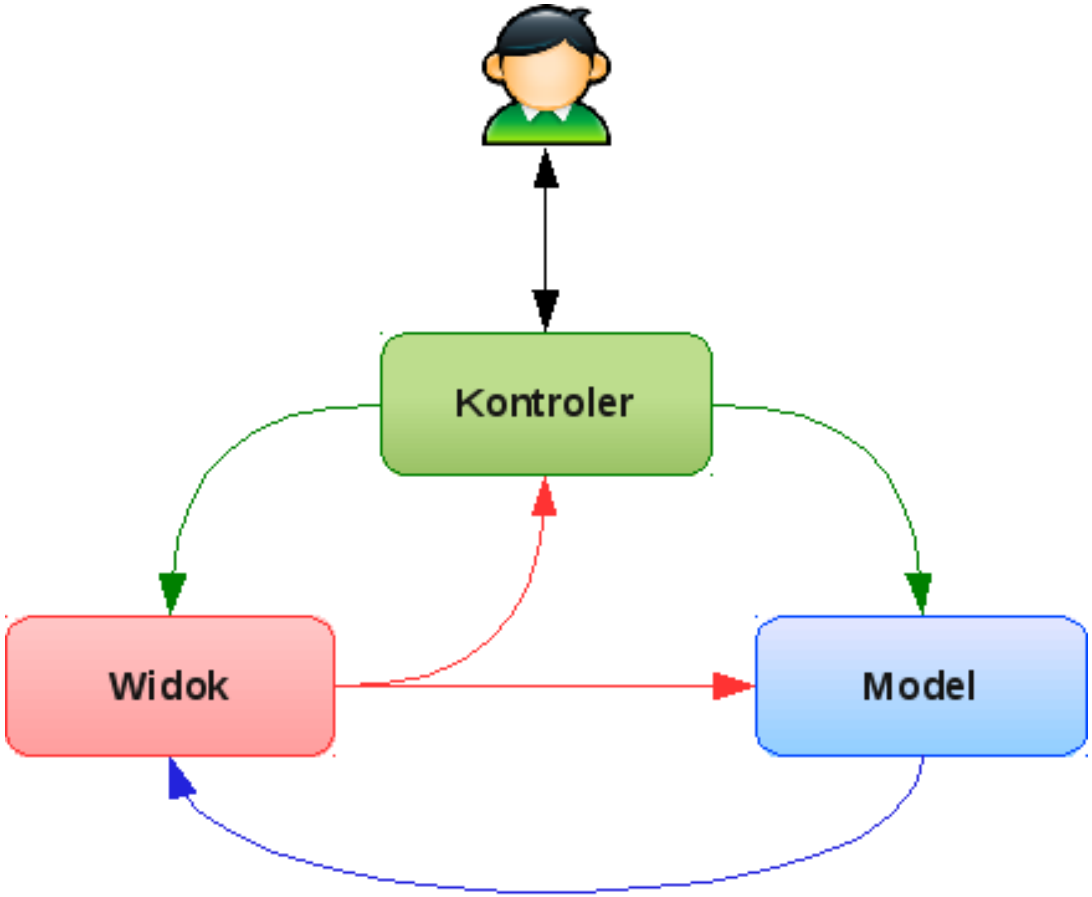
Wzorce kreacyjne umożliwiają opracowanie systemu, którego działanie jest niezależne od tego, jak obiekty są w nim tworzone i przechowywane. Zagadnienie to jest bardzo istotne w momencie, kiedy system oraz wymagania dotyczące systemu zaczynają się zmieniać.

Antywzorce

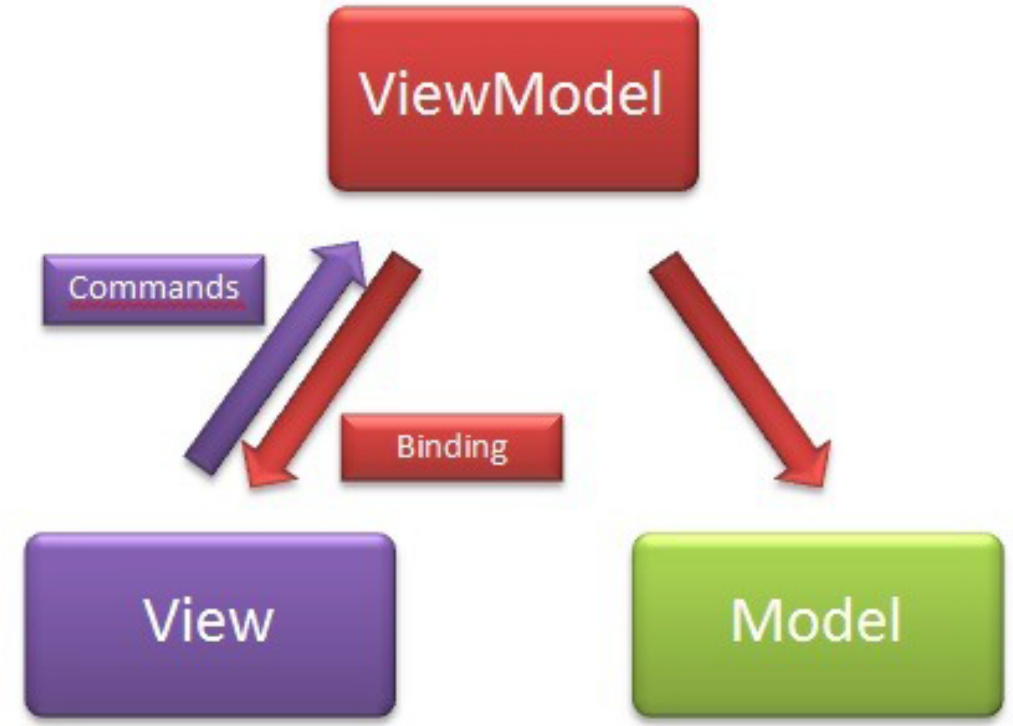
- Mushroom management - całkowite oddzielenie zespołu programistów od klienta (hodowla grzybów);
- nadmiernie rozciągnięta w czasie analiza funkcjonalności systemu;
- nieodpowiednio opracowany plan pracy (nierealne ramy czasowe projektu);
- zbyt duża liczba kluczowych elementów systemu - kamieni milowych;
- zbyt mała liczba kamieni milowych;
- brak zaimplementowanej funkcjonalności;
- system o skomplikowanej, nieczytelnej strukturze - trudny do modyfikowania;
- nieczytelny/zbyt ubogi interfejs użytkownika, gdzie pojedyncza kontrolka odpowiada za zbyt wiele funkcji;
- God object - przerzucenie zbyt dużej liczby funkcjonalności do jednej klasy (rozwiązanie - refaktoryzacja i extract method);

Model - View - Controler

- model - klasy implementujące logikę biznesową (sposób i miejsce przechowywania danych, sposób dostępu do danych);
- widok - logika prezentacyjna (jak dane pochodzące z modeli mają zostać wyświetlone);
- kontroler - logika sterująca całą aplikacją. To właśnie w kontrolerze obsługiwane są wszystkie żądania użytkownika, które następnie delegowane są do odpowiednich metod Modelu.



Rysunek: Model MVC



Rysunek: Model MVVM

Widok

Zadaniem widoku jest wyświetlanie danych – pełni on wyłącznie funkcję prezentacyjną. Nie powinien wykonywać żadnej logiki (biznesowej, czy związanej z przepływem screen'ów).

Model oraz ViewModel

ViewModel stanowi klasę, która eksponuje model dla widoku. Czysty model, zawierający logikę biznesową, prawdopodobnie nie będzie nadawał się do użycia w widoku

Założenia MVVM

- unikanie kodu w code-behind (kod odpowiedzialny za warstwę prezentacji jest oddzielony od kodu odpowiedzialnego za logikę biznesową);
- zdarzenia powinny zostać zastąpione komendami;
- ViewModel powinien implementować interfejs INotifyPropertyChanged;
- dane z widoku powinny być powiązane z właściwościami w ViewModel;

Singleton

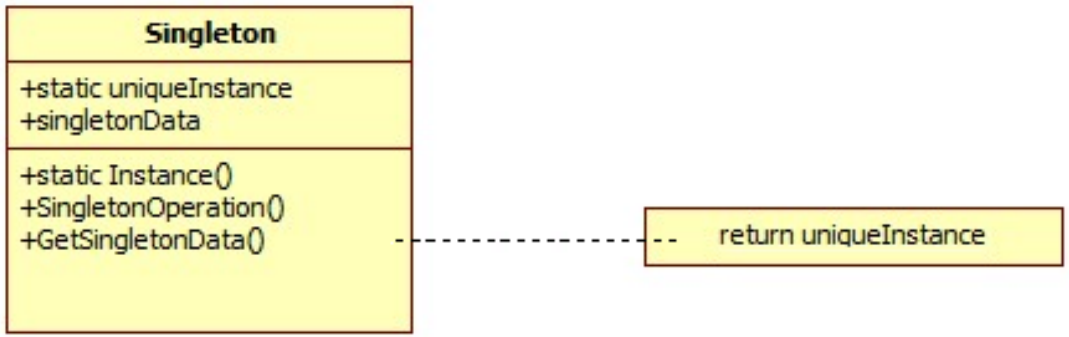
Singleton jest wzorcem konstrukcyjnym, który gwarantuje nam, iż dana klasa będzie miała tylko jeden egzemplarz dostępny globalnie. Przykładem zastosowania tego wzorca jest sytuacja, kiedy to w systemie musimy zagwarantować, iż do obsługi danego elementu - np. menadżera plików, powstanie tylko jeden obiekt.

Singleton - dostęp

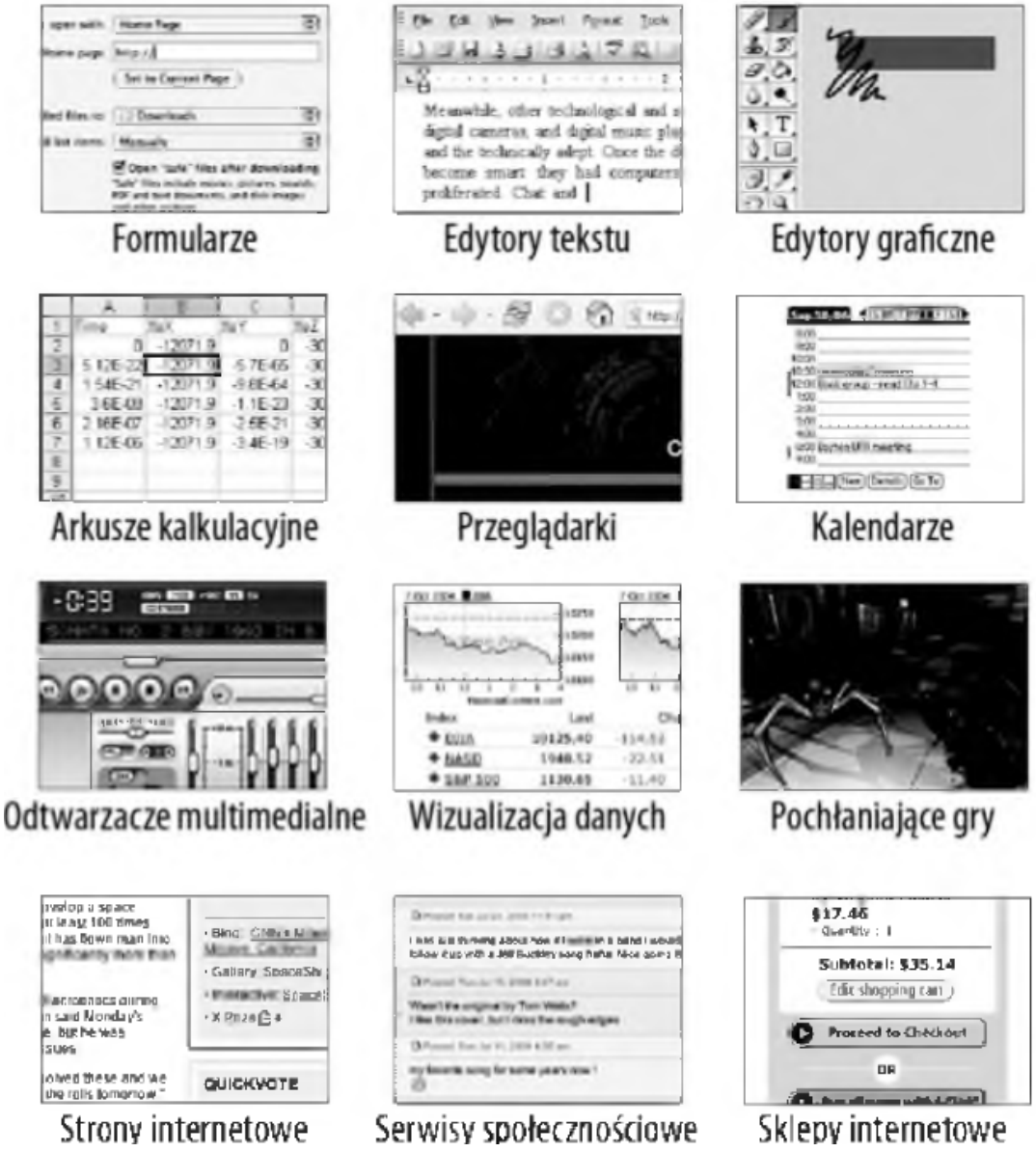
Wewnątrz wzorca definiowana jest statyczna operacja Instance umożliwiająca klientom dostęp do unikalnego egzemplarza klasy. Dostęp do egzemplarza klasy możliwy jest tylko i wyłącznie dzięki metodzie Instance.

Singleton - zalety

- zapewnienie kontroli dostępu do jedynego egzemplarza;
- rozwinięcie koncepcji zmiennych globalnych;
- uminięcie problemu tworzenia wielu zmiennych globalnych.



Rysunek: Singleton - przykład



Rysunek: Zanim przejdziemy do definicji. źródło: Tidwell "Projektowanie interfejsów. Sprawdzone wzorce projektowe"

Wzorce dla interfejsu

- Wzorce to konkretne rozwiązania, a nie ogólne założenia dotyczące aplikacji;
- zachowują zgodność na różnych platformach;
- są gotowymi produktami, a nie procesami;
- są propozycjami, a nie wymaganiami;
- opisują relacje pomiędzy elementami, a nie poszczególne składowe (patrz Windows Navigation Diagram);
- są dostosowane do kontekstów projektowych – w pewnych przypadkach przeniesienie wzorca 1 do 1 ma sens, ale w większości sytuacji konieczna będzie modyfikacja.

14 wzorców (Tidwell J. "Projektowanie interfejsów")

- bezpieczna eksploracja – możliwość swobodnego zapoznania się z interfejsem, ewentualne dostosowanie go do swoich potrzeb;
- pragnienie natychmiastowej satysfakcji – program, który w łatwy sposób pozwala użytkownikowi na wykonanie pewnego działania niejako nagradza go zaraz po uruchomieniu;
- satisficing (satisfying + sufficing) – wystarczająco zadowalający, czyli interfejs, który oferuje pewne możliwości, ale nie wymaga dalszej nauki;
- zmiany na bieżąco – wszystkie formularze i okna dialogowe powinny umożliwiać zapisanie bieżących informacji a także możliwość przerwania danej akcji.

14 wzorców (Tidwell J. "Projektowanie interfejsów")

- odwlekanie decyzji – czyli po prostu wykorzystanie wersji próbnej przed koniecznością upgrade'u do wersji pełnej;
- stopniowa konstrukcja – możliwość budowania menu z pewnego zbioru "cegiełek";
- przyzwyczajenie – najlepiej dostarczyć wraz z interfejsem zestaw poleceń ogólnych, niezależnych od aplikacji. Dlaczego?

Rysunek: Edytor Vi

14 wzorców (Tidwell J. "Projektowanie interfejsów")

- mikroprzerwy – umożliwienie użytkownikowi szybkiego przejścia do wybranej czynności, która nie pochłania czasu (sprawdzenie maila, uruchomienie gry, czytanie e-booka);
- pamięć przestrzenna – określone przyciski powinny być w tym samym miejscu – np. przycisk zamknięcia okna w prawym górnym rogu;
- pamięć prospektywna – przypomnienia, alarmy, powiadomienia wewnątrz aplikacji;
- wspomagane powtarzanie – szybkie wykonanie tego samego polecenia wielokrotnie (np. dla każdego wystąpienia elementu w tekście);
- aplikacja łatwa do sterowania tylko z poziomu klawiatury (analogicznie jak konieczność zmiany sterowania w przypadku gier na urządzenia mobilne);

14 wzorców (Tidwell J. "Projektowanie interfejsów")

- porady innych – lista opcji (np. przedmiotów do zakupienia) wyświetlana na podstawie preferencji innych użytkowników – interfejsy z wykorzystaniem systemu rekomendacji. Zarówno dla serwisów z zakupami, jak i sugerowanego układu przycisków przy pierwszym uruchomieniu aplikacji.
- osobiste rekomendacje – czyli możliwość współdzielenia informacji przy pomocy mediów społecznościowych (poprzez np. udostępnienie linka).

Modele nawigacji

Jak wygląda połączenie głównych komponentów aplikacji (Frame'y, okna, panele, widoki) i jakie są zależności pomiędzy nimi. W przypadku aplikacji desktopowej często jest to ograniczone wykorzystanym wzorcem np. (MVC), który narzuca pewne powiązania poprzez wykorzystanie kontrolera. Dobrze widać to przy projektowaniu bazującym na storyboarding.

- Nawigacja globalna – treść i materiały znajdujące się na stronie głównej aplikacji, lub strony internetowej. Najczęściej jest to menu główne, menu boczne, pasek wyszukiwania, chmurki tagów;
- Nawigacja funkcyjna – niższego poziomu, związana bezpośrednio z danym elementem menu, z danym elementem strony. Przykładowo zakładka kontakt może udostępniać dodatkowy widok kalendarza (którzy połączony jest z kalendarzem google, który to z kolei połączony jest zarówno z naszym telefonem, jak i aplikacją desktopową typu PIM).

Architektura i jej rodzaje

- oś i szprychy – najczęściej stosowana w aplikacjach mobilnych, gdzie jeden centralny widok prowadzi do poszczególnych elementów składowych;
- pełne połączenie – spotykane często w przypadku witryn internetowych, gdzie możliwość przejścia z jednej zakładki / podstrony, do innego (dowolnego komponentu) możliwa jest poprzez jedną interakcję z interfejsem;
- wiele poziomów – rzadziej spotykany układ stron, gdzie główne elementy są ze spobą połączone podobnie jak we wcześniejszech architekturze, natomiast na pewnym poziomie zagnieżdżenia przejście (powrót) do strony nadrzędnej nie jest możliwy;

Architektura i jej rodzaje

- krok po kroku – pokaz slajdów (wyjątek stanowi tutaj np. Prezi, gdzie mamy możliwość tworzenia rozgałęzień w prezentacji). Zasada prezentacji jest jednak taka, żeby odbiorca miał możliwość zapoznawania się ze wszystkimi udostępnionymi dla niego elementami;
- piramida – wariant poprzedniej architektury rozwinięty o dodatkowy element widoku nadrzędnego, który umożliwia przemieszczanie się pomiędzy poszczególnymi widokami;
- przeciąganie i przybliżanie – mapy, obrazy, duże dokumenty tekstowe, które przegląda się tylko fragmentami. Dodatkowymi elementami w tym przypadku mogą być przyciski widoczne w trakcie przeglądania, ale nie połączone na stałe z komponentem (nie przesuwające się razem z mapą).

Wzorzec – obszar centralny

- stosowany, kiedy jeden z elementów aplikacji ma pełnić kluczową funkcję;
- najbardziej oczywisty przykład takiego wzorca to edytory tekstu, gdzie cały środkowy obsar ekranu zajmuje biały obszar imitujący papier;
- podobna funkcja obszaru centralnego stosowana jest w środowiskach programistycznych, gdzie dodatkowo istotnym obszarem jest część przeznaczona na Output aplikacji;
- programy graficzne mają również wyszczególniony obszar centralny, jednak w tym przypadku najczęściej jest on otoczony dodatkowymi komponentami, zatem nacisk przesuwa się nieco w kierunku wzorca "siatki równoprawnych elementów".

Rysunek: Wzorzec wyważenia po przekątnej – gdzie narożniki dociążone są pod względem liczby elementów

Architektura informacji

- jak długi jest zasób (lista) do wyświetlenia;
- czy wyniki pochodzą z wyszukiwarki?
- czy elementy powinny być posortowane? (długa lista elementów bez sortowania praktycznie uniemożliwia działanie);
- czy użytownikowi udostępniony jest mechanizm sortowania listy?
- czy wyświetlane elementy powinny być pogrupowane?
- jakie są kategorie nadrzędne wyświetlanych elementów / czy istnieje więcej sposobów pogrupowania elementów?
- czy elementy listy różnią się typem?
- czy lista jest statyczna, czy też dynamiczna?

Architektura informacji II

- jak często zmianie ulegają elementy na liście dynamicznej?
- czy wyświetlanie elementów zależy od użytkownika? (wyświetlanie w zależności od wybranego konta);
- czy elementy powinny być w jakiś sposób rekomendowane użytkownikowi?
- co z użytkownikiem, który używa aplikacji po raz pierwszy (problem zimnego startu);
- czy rekomendować elementy użytkownikowi na podstawie danych z innych kont?

Dziękuję za uwagę