Modelowanie z wykorzystaniem notacji UML

Część 2 - diagramy stanów i klas (miniprojekt)

Temat zajęć

Celem zajęć jest wykorzystanie przez studentów wiedzy o UML zdobytej na ostatnich zajęciach do zaprojektowania diagramu klas oraz stanów do miniprojektów. Jednocześnie, materiały zawierają informacje odnośnie kilku ważnych koncepcji stosowanych przy projektowaniu architektury różnych systemów.

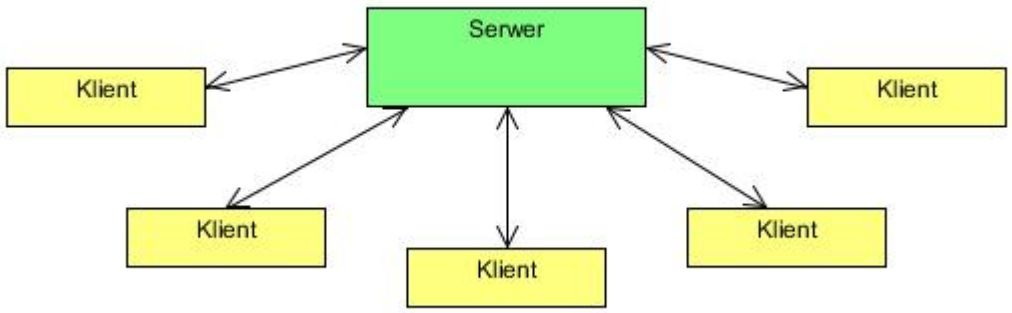
Popularne koncepcje architektoniczne

Wprowadzenie

Oczywiście, każde oprogramowanie jest na tyle specyficzne, iż wymaga "spersonalizowanego" podejścia do projektu struktury i modelu obiektowego. Jednakże, warto poznać pewne sztandardowe koncepcje stosowane z sukcesami w większości obecnych systemów.

Warto pamiętać o tym, że programowanie obiektowe pozwala na łatwiejsze wprowadzenie modułowości do systemu. Ta cecha jest istotna z co najmniej paru powodów. Po pierwsze, pozwala na łatwiejszą modyfikację konkretnego fragmentu kodu, nie naruszając przy tym reszty programu. Po drugie, w przypadku aktualizacji i odpowiedniej architektury, można wymieniać konkretne moduły bez konieczności wyłączania innych. Po trzecie, dzięki takiemu podejściu, istnieje możliwość (w zależności od dodatkowych czynników) napisania każdego modułu w innej technologii, co w niektórych przypadkach może być pożądane.

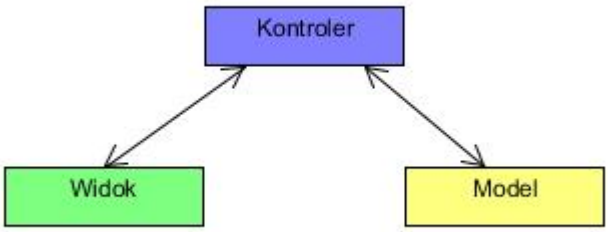
Klient-serwer



Podejście klient-serwer jest niezwykle często stosowane przy wszelkich aplikacjach sieciowych, gdzie istnieje pewna aplikacja zwana **serwerem**, odbierająca żądania od klientów, przetwarzająca dane i odsyłająca odpowiedź. **Aplikacji klienckich** może być wiele

i komunikują się z serwerem w celu pozyskania pewnych danych lub ich synchronizacji. Klasycznym przykładem podejścia klient-serwer jest wyświetlenie strony internetowej (umieszczonej na serwerze) przy pomocy przeglądarki internetowej (klient). Alternatywnym podejściem do klient-serwer jest **Peer-to-Peer (P2P)**, gdzie każda z uczestniczących w procesie stron jest równoważna i może przyjmować jednocześnie rolę klienta jak i serwera.

Model-widok-kontroler (MVC)



To kolejna koncepcja bardzo często wykorzystywana w aplikacjach internetowych. Polega na wyraźnym oddzieleniu od siebie trzech modułów: wyświetlającego (**widok**), dostępu do danych (**model**) oraz pośredniczącej (**kontroler**), przy czym jedynie kontroler może komunikować się z pozostałymi warstwami, natomiast model i widok nie mają bezpośredniego połączenia. MVC jest koncepcją praktycznie tożsamą z pojęciem **architektury trójwarstwowej**, choć zwykle ten drugi typ jest uznawany za bardziej ogólny. Z innego punktu widzenia, ten schemat można opisać na przykładzie gry - model przechowuje jednostki, które aktualnie znajdują się w grze. Kontroler pobiera je i przekazuje do widoków, którymi są różne rodzaje mapy (główna oraz minimapa), a także panele wyświetlające aktualnie posiadane przez gracza jednostki.

Oprogramowanie do tworzenia diagramów UML

Na początku należy wspomnieć, iż zupełnie wstępne projekty diagramów często przygotowuje się w sposób odręczny, na kartce. Trzeba pamiętać, iż przy manualnym tworzeniu często łatwiej się skupić i myśleć kreatywnie. Jednakże, zazwyczaj dużo atrakcyjniejszym i łatwiejszym w modyfikacji i prezentacji rozwiązaniem jest użycie jednego z programów służących do tworzenia diagramów UML. Poniżej znajduje się zestawienie paru wybranych programów:

- UMLet (Java)
- StarUML (Windows/OSX)
- Astah (Java)
- Dia (Linux)
- Lucidchart (WWW)
- Yuml (WWW)
- Microsoft Visio (Windows)
- Visual Paradigm (Cross-platform)

Wszystkie diagramy w tym dokumencie (oraz w dokumencie do poprzednich zajęć) zostały stworzone przy pomocy całkowicie darmowego programu UMLet.

Zadanie na zajęcia

Zadaniem studentów na zajęciach jest rozpoczęcie prac nad modelem obiektowym do własnego miniprojektu. Należy przy tym uwzględnić wszystkie wytworzone wcześniej artefakty - zakres projektu, aktorów, obiekty biznesowe, zależności i założenia, wymagania funkcjonalne czy wymagania pozafunkcjonalne.

Oczekuje się wykonania dwóch zestawów diagramów:

- diagramu klas obejmującego wszystkie wymagania funkcjonalne (i inne aspekty) wymienione w specyfikacji wymagań
- diagramów stanów dla wyspecyfikowanych obiektów biznesowych

Z uwagi na prawdopodobnie dość dużą wielkość diagramu klas, dopuszcza się podzielenie schematu na części. Dla każdego obiektu biznesowego powinien zostać zawarty osobny diagram stanów. Przy wykonywaniu ćwiczeń należy trzymać się wskazówek dotyczących notacji UML z poprzednich zajęć.

W przesłanych plikach należy również uwzględnić opis diagramów, aby prowadzący mógł łatwo się w nich zorientować. Nie dopuszcza się również przesyłania prac w formatach specyficznych dla programów do tworzenia diagramów UML - należy przekazać wersje wyeksportowane (np. do pliku graficznego, PDF czy HTML).

Termin na wysłanie prac prowadzącemu przypada na koniec dnia (23:59:59) po 2 tygodniach od zajęć, na których omawiany jest temat. Zadanie oceniane jest maksymalnie na 15 punktów, a za każdy dzień opóźnienia odejmowane jest 0,5 punkta od końcowego rezultatu.