Kurs programowania

Wykład 12

Wojciech Macyna

2 czerwca 2016

Czym jest UML?

UML składa się z dwóch podstawowych elementów:

- notacja: elementy graficzne, składnia języka modelowania,
- metamodel: definicje pojęć języka i powiązania pomiędzy nimi

Z punktu widzenia modelowania ważniejsza jest notacja.
Z punktu widzenia generacji kodu – metamodel.

Sposoby patrzenia na modelowanie

Perspektywa przypadków użycia

Opisuje funkcjonalność, jaką powinien dostarczać system, widzianą przez jego użytkowników.

Perspektywa logiczna

Zawiera sposób realizacji funkcjonalności, strukturę systemu widzianą przez projektanta.

Perspektywa implementacyjna

Opisuje poszczególne moduły i ich interfejsy wraz z zależnościami; perspektywa ta jest przeznaczona dla programisty.

Sposoby patrzenia na modelowanie

Perspektywa przypadków użycia

Opisuje funkcjonalność, jaką powinien dostarczać system, widzianą przez jego użytkowników.

Perspektywa logiczna

Zawiera sposób realizacji funkcjonalności, strukturę systemu widzianą przez projektanta.

Perspektywa implementacyjna

Opisuje poszczególne moduły i ich interfejsy wraz z zależnościami; perspektywa ta jest przeznaczona dla programisty.

Sposoby patrzenia na modelowanie

Perspektywa przypadków użycia

Opisuje funkcjonalność, jaką powinien dostarczać system, widzianą przez jego użytkowników.

Perspektywa logiczna

Zawiera sposób realizacji funkcjonalności, strukturę systemu widzianą przez projektanta.

Perspektywa implementacyjna

Opisuje poszczególne moduły i ich interfejsy wraz z zależnościami; perspektywa ta jest przeznaczona dla programisty.

Sposoby patrzenia na modelowanie

Perspektywa procesowa

Zawiera podział systemu na procesy (czynności) i procesory (jednostki wykonawcze); opisuje właściwości pozafunkcjonalne systemu i służy przede także programistom i integratorom.

Perspektywa wdrożenia

Definiuje fizyczny podział elementów systemu i ich rozmieszczenie w infrastrukturze; perspektywa taka służy integratorom i instalatorom systemu.

Sposoby patrzenia na modelowanie

Perspektywa procesowa

Zawiera podział systemu na procesy (czynności) i procesory (jednostki wykonawcze); opisuje właściwości pozafunkcjonalne systemu i służy przede także programistom i integratorom.

Perspektywa wdrożenia

Definiuje fizyczny podział elementów systemu i ich rozmieszczenie w infrastrukturze; perspektywa taka służy integratorom i instalatorom systemu.

Rodzaje diagramów UML

- Diagram przypadków użycia
- Diagram pakietów
- Diagram klas
- Diagram strukturalny
- Diagram komponentów
- Diagram wdrożenia
- Diagram stanów
- Diagram aktywności (czynności)
- Diagramy interakcji (współpracy, przebiegu (sekwencji), komunikacji, przeglądu interakcji)
- Diagram przebiegów czasowych

Diagram przypadków użycia

Diagram przypadków użycia opisuje system z punktu widzenia użytkownika, pokazuje co robi system, a nie jak to robi. Diagram ten sam w sobie zazwyczaj nie daje nam zbyt wielu informacji, dlatego też zawsze potrzebna jest do niego dokumentacja w postaci dobrze napisanego przypadku użycia. Przypadki użycia są bardzo ważnym narzędziem zbierania wymagań. Diagramy przypadków użycia, mimo swojej prostoty, są bardzo przydatne, gdyż tworzą swojego rodzaju spis treści dla wymagań modelowanego systemu.

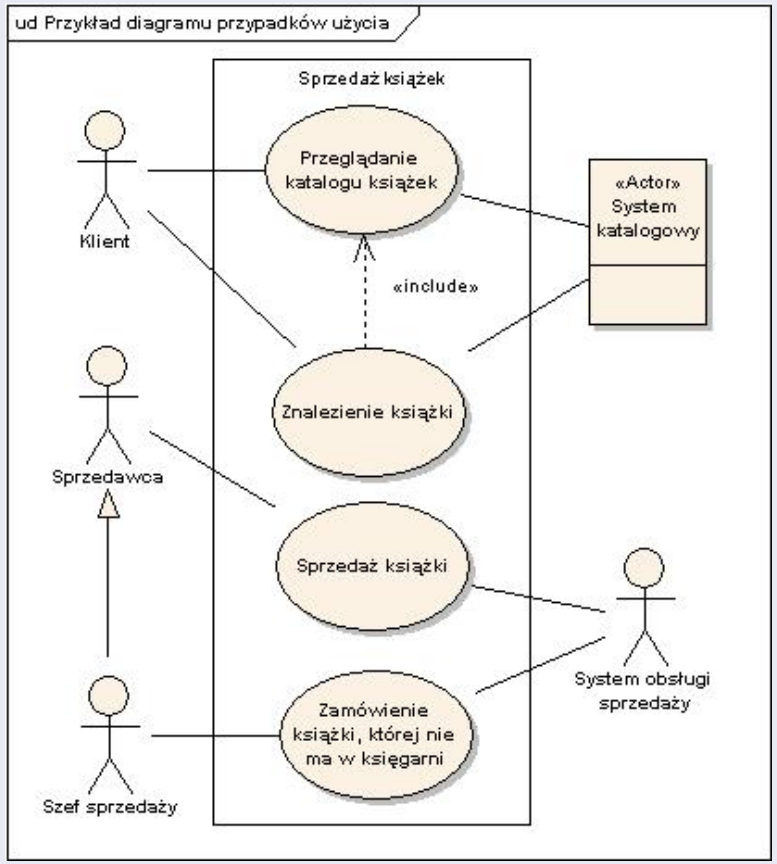


Diagram pakietów

Diagram pakietów służy do tego, by uporządkować strukturę zależności w systemie, który ma bardzo wiele klas, przypadków użycia itp. Przyjmujemy, że pakiet zawiera w sobie wiele elementów, które opisują jakieś w miarę dobrze określone zadanie. Na diagramie umieszczamy pakiety i wskazujemy na zależności między nimi. Dzięki temu dostajemy na jednym diagramie obraz całości, bądź dużego fragmentu, systemu.

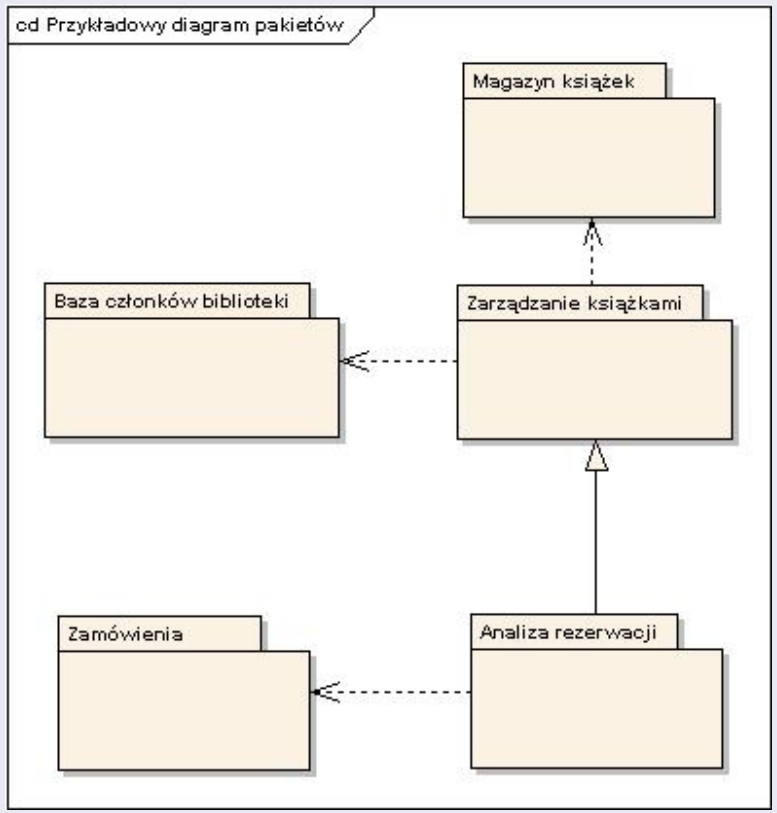


Diagram klas

Diagram klas jest ściśle powiązany z projektowaniem obiektowym systemu informatycznego lub wręcz bezpośrednio z jego implementacją w określonym języku programowania. Elementami tego diagramu są klasy, reprezentowane przez prostokąty, które mogą zawierać informację o polach i metodach klasy.

UML definiuje 4 poziomy widoczności cech i metod: + publiczny, # chroniony (klasa i jej podklasy), – prywatny, ~ publiczny wewnątrz pakietu.

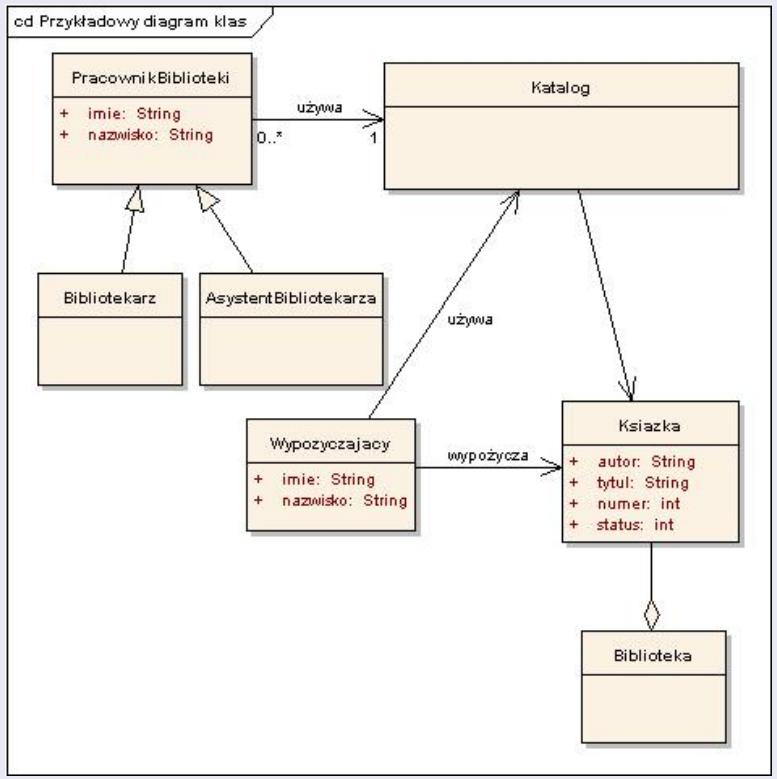
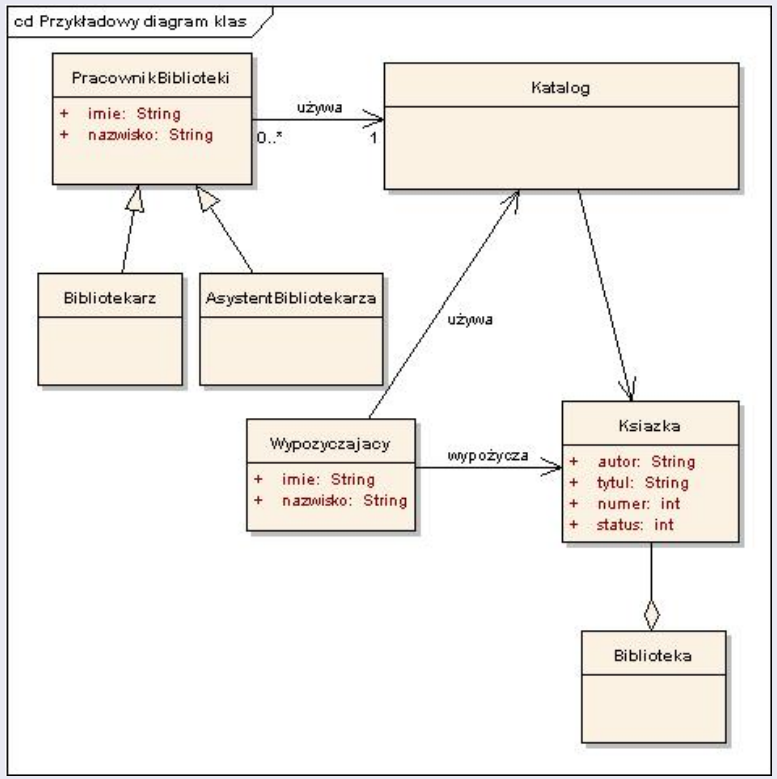


Diagram klas

Diagram klas jest ściśle powiązany z projektowaniem obiektowym systemu informatycznego lub wręcz bezpośrednio z jego implementacją w określonym języku programowania. Elementami tego diagramu są klasy, reprezentowane przez prostokąty, które mogą zawierać informację o polach i metodach klasy.



UML definiuje 4 poziomy widoczności cech i metod: + publiczny, # chroniony (klasa i jej podklasy), – prywatny,˜ publiczny wewnątrz pakietu.

Diagram strukturalny

Jest przeznaczony do tego, by modelować współpracę klas, interfejsów, komponentów, które są zaangażowane w pewne zadanie. Diagram ten jest nieco podobny do diagramu klas, z tą różnicą, że diagram klas przedstawia statyczny obraz fragmentu systemu, a diagram strukturalny obrazuje elementy systemu wykonujące wspólne zadanie, typowe sposoby użycia elementów systemu, związki między tymi elementami, które może być trudno przedstawić na innych diagramach.

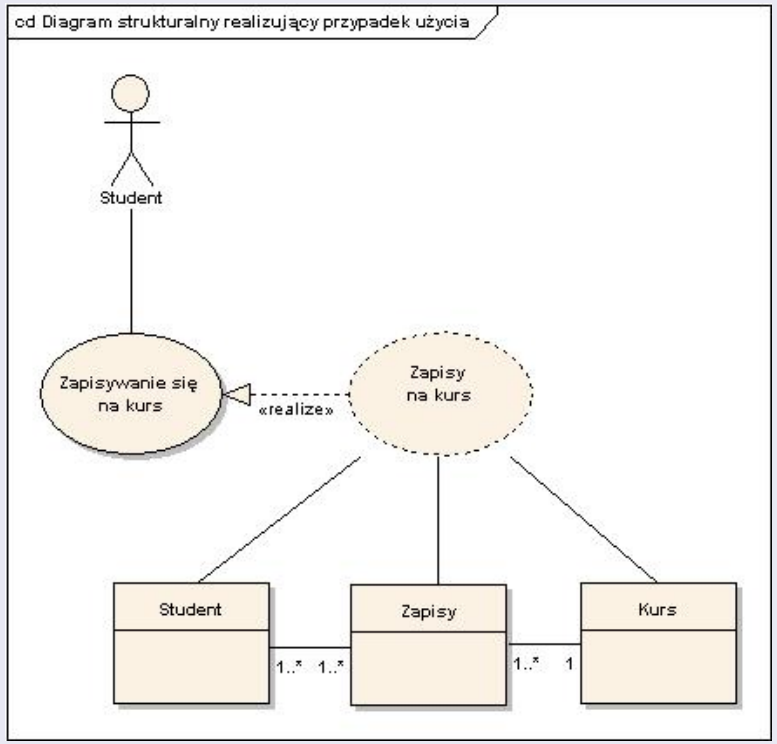


Diagram komponentów

Diagram komponentów robimy z podobnych powodów, co diagram pakietów – chcemy podzielić system na prostsze elementy i pokazać zależności między nimi. Diagram pakietów koncentrował się na podziale systemu z logicznego punktu widzenia, diagram komponentów z kolei dzieli system na fizyczne elementy oprogramowania: pliki, biblioteki, gotowe, wykonywalne programy itp.

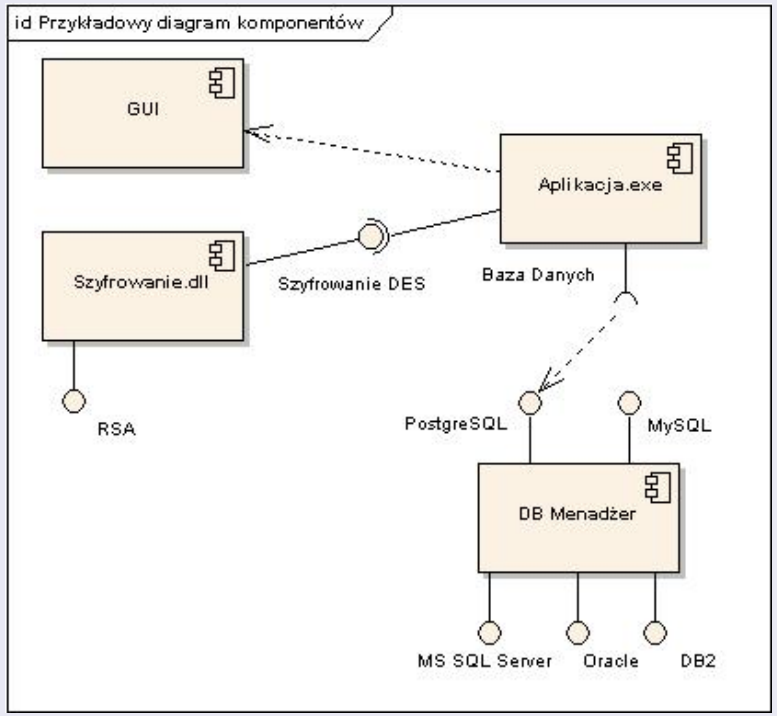


Diagram wdrożenia

Diagram wdrożenia pokazuje, jak będzie wyglądało wdrożenie i konfiguracja naszego oprogramowania.

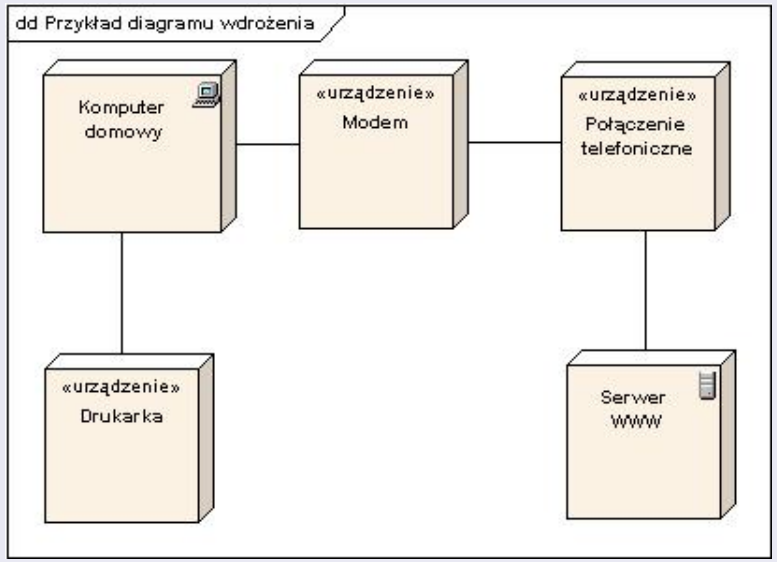


Diagram stanów

Diagram stanów z służy do tego, by pokazać w jakich stanach mogą być obiekty.

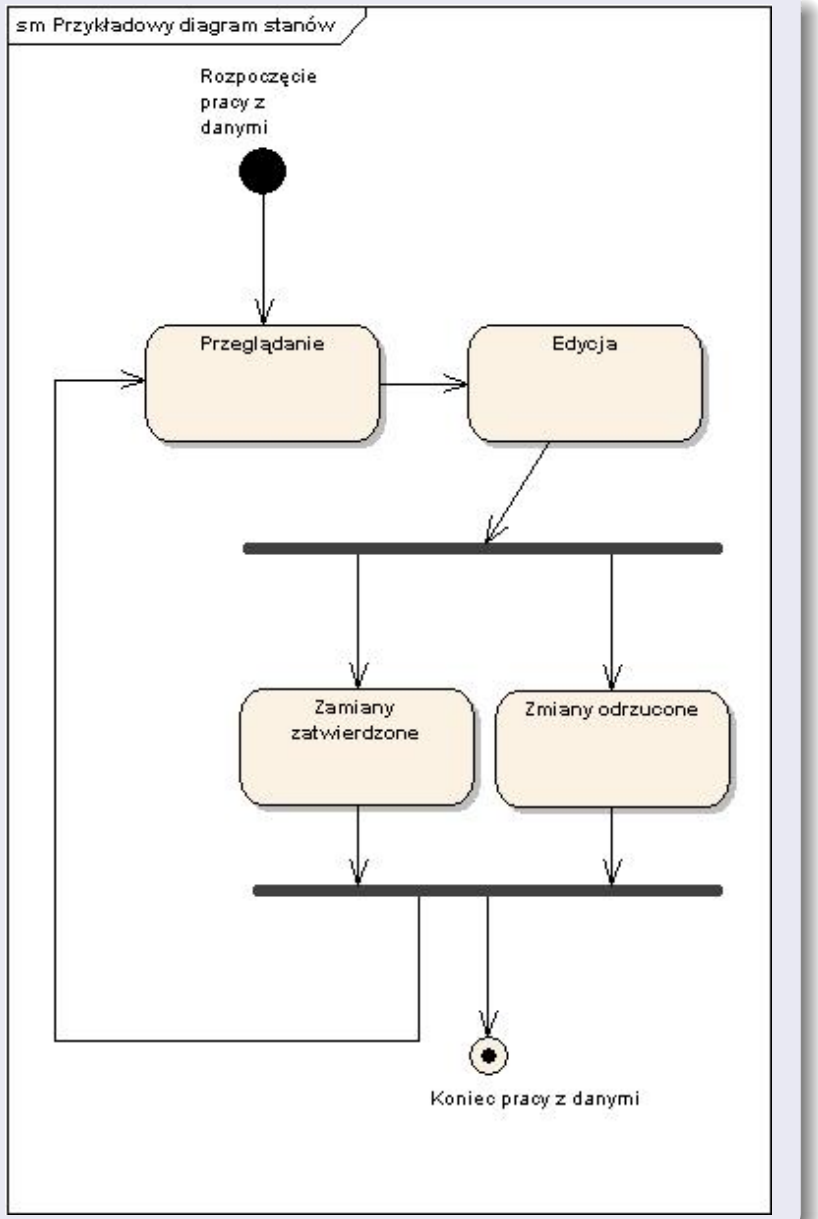


Diagram aktywności

Diagram aktywności jest pewną mutacją diagramu stanów, z tą różnicą, że diagram aktywności skupia się raczej na opisaniu jakiegoś procesu, w którym uczestniczy wiele obiektów, zaś diagram stanów pokazuje, jakie są możliwe stany konkretnego obiektu. Diagram aktywności jest bardzo dobrym narzędziem, gdy chcemy przedstawić odpowiedzialność obiektów w ramach jakiegoś procesu.

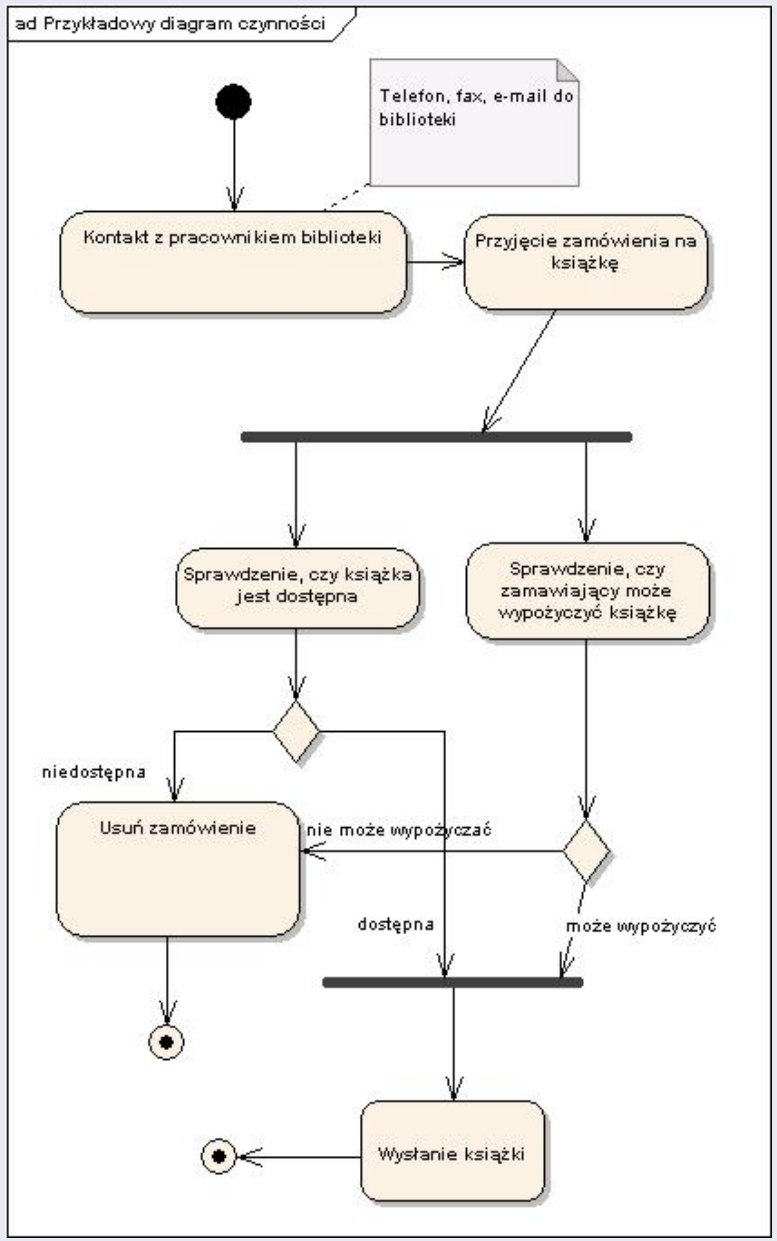


Diagram współpracy (komunikacji)

Diagram współpracy jest jednym z diagramów interakcji. Używamy go po to, żeby zobrazować dynamikę systemu – wzajemne oddziaływanie na siebie obiektów oraz komunikaty, jakie między sobą przesyłają.

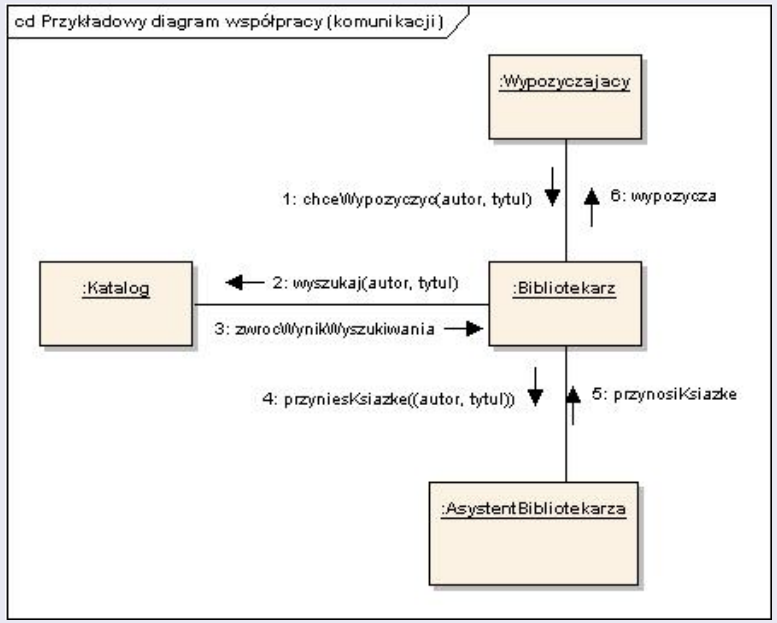


Diagram przebiegu (sekwencji)

Analogiczną informację do diagramu komunikacji zawiera drugi z diagramów interakcji, diagram przebiegu. Diagram komunikacji koncentrował się na zobrazowaniu współpracy między obiektami, teraz chcemy pokazać kolejność przesyłania komunikatów i czas istnienia obiektów.

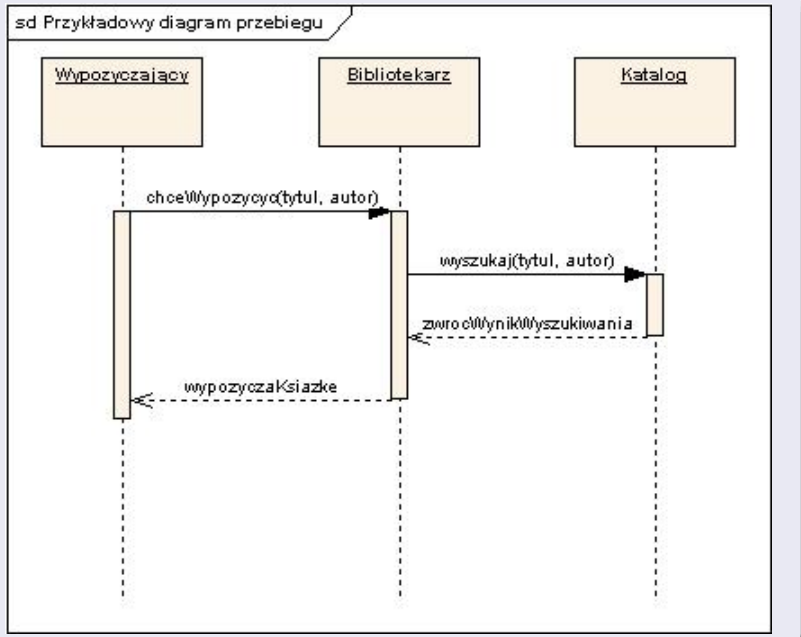
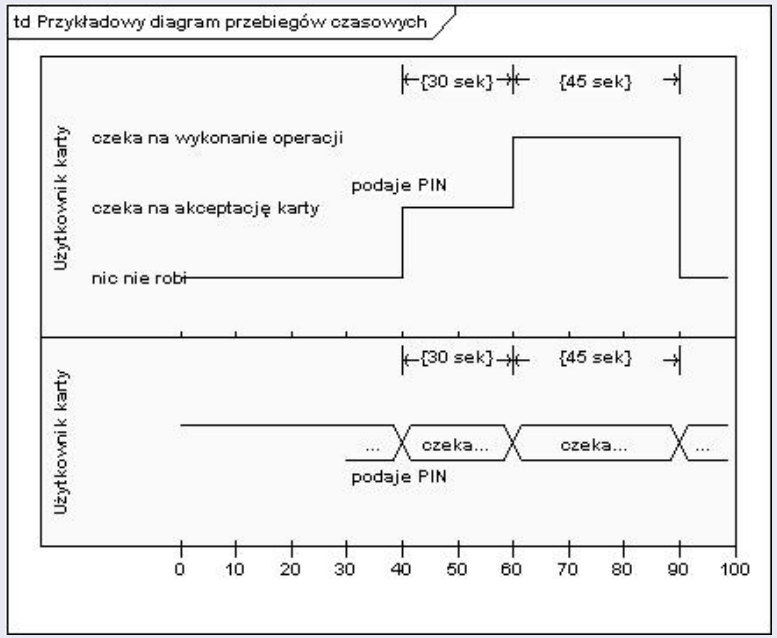


Diagram przebiegów czasowych

Diagram przebiegów czasowych obrazuje zachowanie obiektu z naciskiem na dokładne określenie czasu, w którym obiekt jest poddawany jakimś zmianą lub sam wykonuje jakieś działanie.



Podsumowanie

- Diagramy komponentów i wdrożenia przedstawiają logiczną i fizyczną strukturę podsystemów.
- Diagramy interakcji służą do opisu komunikacji pomiędzy obiektami
- Diagramy czynności definiują algorytmy realizacji funkcji, a diagramy stanu – zmianę zachowania obiektów.

Bardziej szczegółowe informacje na kursie *Technologia Programowania* (III semestr).

Podsumowanie

- Diagramy komponentów i wdrożenia przedstawiają logiczną i fizyczną strukturę podsystemów.
- Diagramy interakcji służą do opisu komunikacji pomiędzy obiektami
- Diagramy czynności definiują algorytmy realizacji funkcji, a diagramy stanu – zmianę zachowania obiektów.

Bardziej szczegółowe informacje na kursie *Technologia Programowania* (III semestr).