

Inżynieria oprogramowania i analiza biznesowa

Wykład 5: Unified Modeling Language: Diagramy interakcji i diagramy stanów Diagramy fizyczne

Marek Krętowski
pokój 13a
e-mail: m.kretowski@pb.edu.pl
<http://aragorn.pb.bialystok.pl/~mkret>



Wersja 1.4 st. podyplomowe

Diagramy interakcji

Diagram interakcji - obrazuje interakcję jako zbiór obiektów i związków między nimi, w tym też komunikaty, które obiekty przekazują między sobą; zawierają na ogół obiekty, wiązania oraz komunikaty; cel: modelowanie przepływu sterowania

Diagram **współpracy**, kooperacji (ang. collaboration) - uwypukla związki strukturalne pomiędzy obiektami (organizacja strukturalna) wysyłającymi i odbierającymi komunikaty; graficznie jest to zestaw wierzchołków i krawędzi

- nacisk na związki strukturalne między egzemplarzami uczestniczącymi w interakcji oraz komunikaty przesyłane między nimi
- wygodniejsze do przedstawiania złożonych iteracji i rozgałęzień
- stosowane przy wielu współbieżnych przepływach sterowania

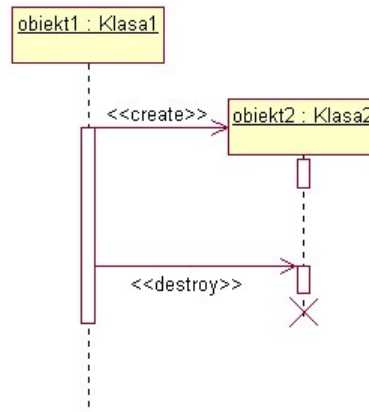
Diagram **przebiegu** (ang. sequence) - uwypukla kolejność komunikatów w czasie; ma postać tabeli, w której obiekty są ułożone wzdłuż osi X a komunikaty wzdłuż osi Y, uporządkowane wg czasu ich wysłania

- nacisk na sposób przekazywania komunikatów w miarę ich pojawiania się
- szczególnie przydatne w kontekście scenariusza przypadków użycia
- wygodniejsze do przedstawiania prostych iteracji i rozgałęzień

Diagramy przebiegu

Dwie podstawowe cechy:

- **Linia życia obiektów** - pionowe przerywane kreski reprezentujące czas istnienia obiektów (zwykle obiekty żyją przez cały czas trwania interakcji, wpp od odebrania komunikatu stereotypowego <<create>> aż do otrzymania <<destroy>>)
- **Ośrodek sterowania** (ang. *focus of control*) - podłużny, cienki prostokąt reprezentujący okres wykonywania przez obiekt akcji - osobiście lub przez procedury podrzędne; zagnieżdżenia sterowania (np. wywołania rekurencyjne lub wywołania własnych operacji) oznaczane są za pomocą dodatkowego prostokąta ośrodka sterowania umieszczonego na prawo od jego przodka

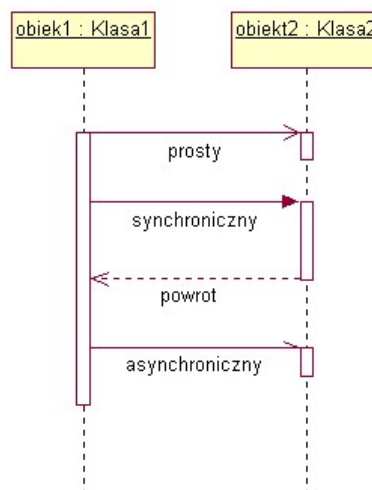


Inżynieria oprogramowania (Wyk. 5)

Slajd 3 z 34

Rodzaje komunikatów (UML 1.5)

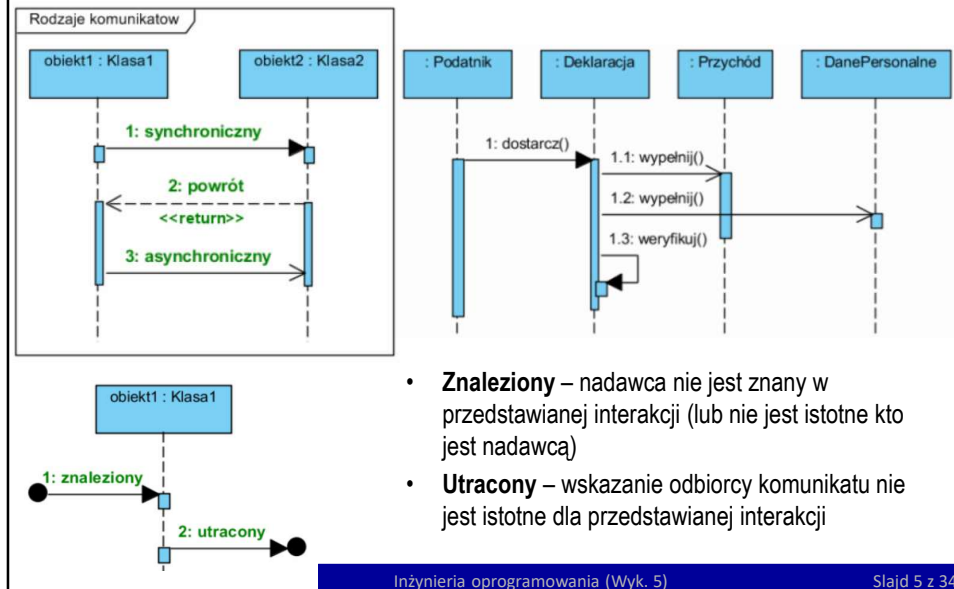
- **Prosty** - powoduje jedynie przekazanie sterowania od obiektu do obiektu; na danym etapie modelowania nie są istotne szczegóły tego przekazania
- **Synchroniczny** - obiekt wysyłający komunikat oczekuje na odpowiedź zwrotną i dopiero po jej otrzymaniu przechodzi do dalszych działań; zwykle reprezentuje wywołanie proceduralne
- **Powrót** - oznacza powrót z wywołania procedury; może być pomijany (i najczęściej jest), gdyż jest nieuchronną konsekwencją wywołania
- **Asynchroniczny** - po wysłaniu komunikatu obiekt kontynuuje swoje działania bez oczekiwania na odpowiedź



Inżynieria oprogramowania (Wyk. 5)

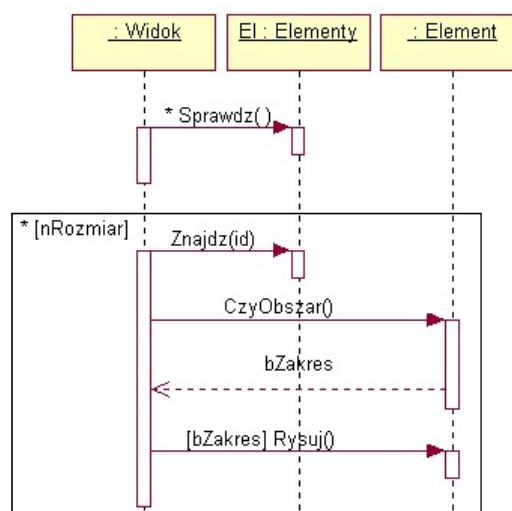
Slajd 4 z 34

Rodzaje komunikatów (UML 2.X)



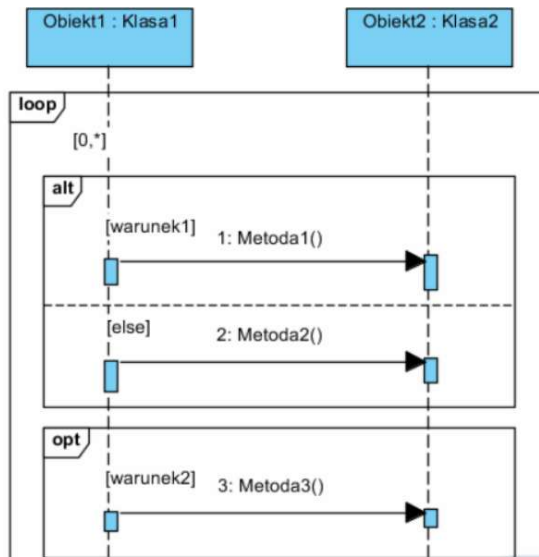
Iteracja (UML 1.5)

- Stosowana w celu zobrazowania ciągu powtarzających się komunikatów
- Zaznaczana na diagramie wyrażeniem iteracyjnym przed nazwą komunikatu:
 - * - ("sama gwiazdka" - liczba powtórzeń nie jest określona)
 - * [i:=1..n]
- W sytuacji gdy iteracja dotyczy kilku komunikatów, obejmowane one są w ramkę i wyrażenie iteracyjne podawane jest w lewym górnym rogu



Ramki (UML 2.0)

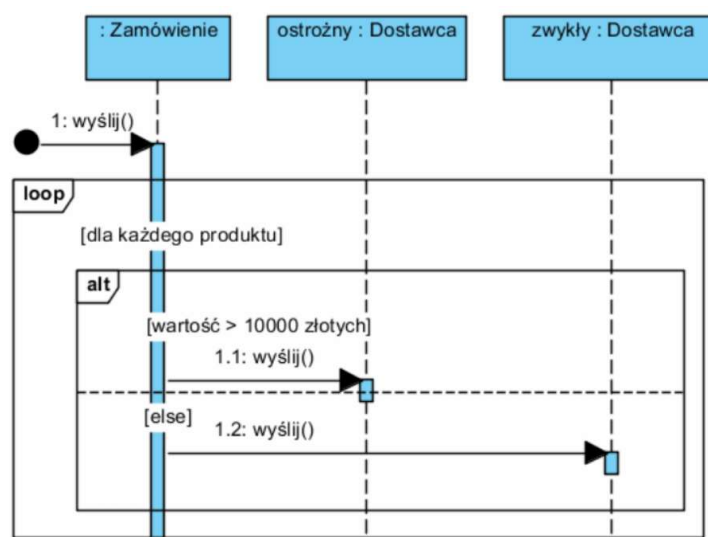
- ALT – alternatywa; wykonywany jest tylko ten fragment, przy którym spełniony jest warunek
- OPT – opcjonalność; wykonywany jeśli warunek jest spełniony
- LOOP – pętla; warunek określa podstawę iteracji
- REF – odwołanie; stanowi referencję do innej interakcji; przykrywa linie życia
- PAR – współbieżność; wszystkie fragmenty wykonywane są współbieżnie
- Wiele innych (np. CRITICAL, NEG, ...)



Inżynieria oprogramowania (Wyk. 5)

Slajd 7 z 34

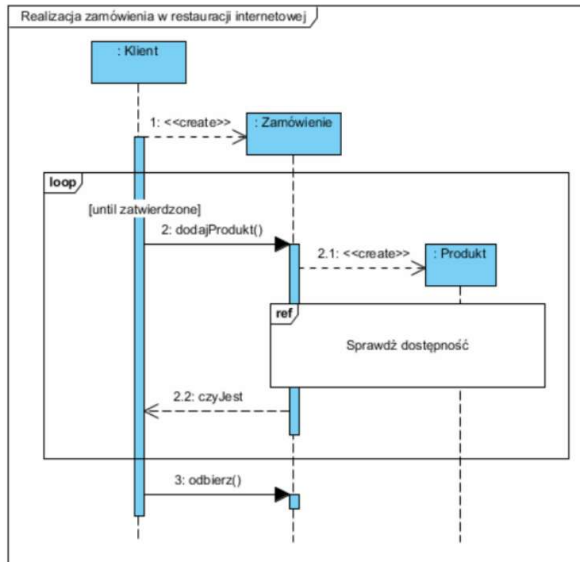
Ramki – przykłady (1)



Inżynieria oprogramowania (Wyk. 5)

Slajd 8 z 34

Ramki – przykłady (2)

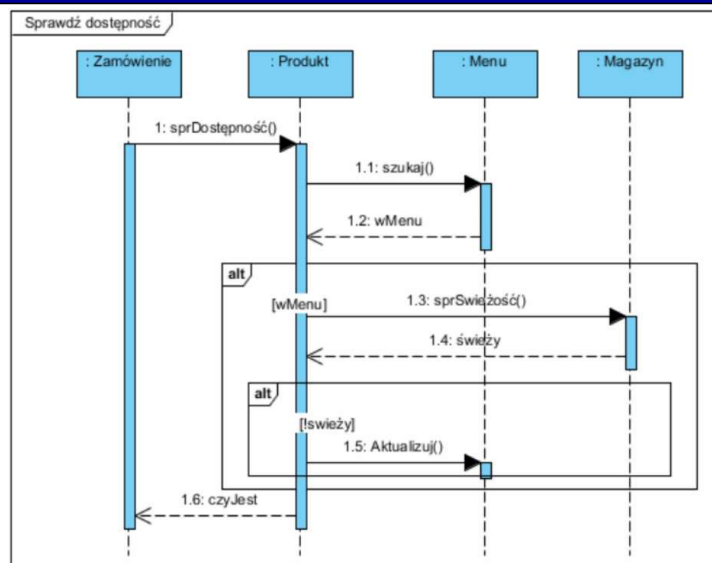


- Ramka REF jest de facto oddzielnym diagramem (bardziej szczegółowym)
- „Sprawdź dostępność” - przedstawiona na kolejnym slajdzie

Inżynieria oprogramowania (Wyk. 5)

Slajd 9 z 34

Ramki – przykłady (3)

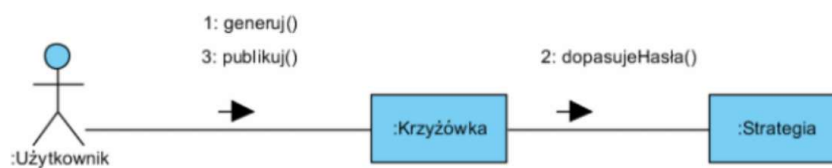


Inżynieria oprogramowania (Wyk. 5)

Slajd 10 z 34

Diagramy współpracy

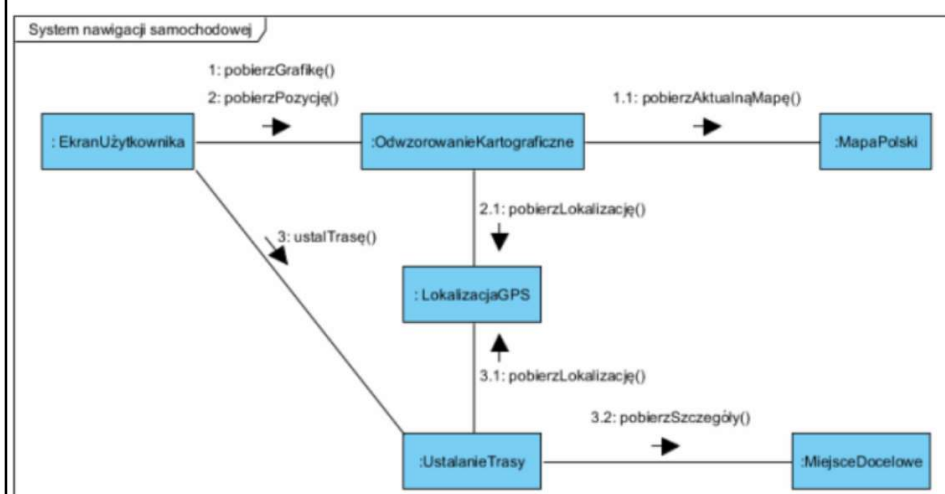
- Uwaga skupiona na tym w jaki sposób obiekty ze sobą współpracują poprzez wymianę/przesyłanie komunikatów
- Umożliwia pokazanie powiązań stanowiących kontekst
- W celu podania kolejności komunikatów w czasie niezbędne jest ich ponumerowanie (notacja Deweya umożliwia przedstawienie zagnieżdżeń, np. 1,1.1, 1.2, 2, ...)



Inżynieria oprogramowania (Wyk. 5)

Slajd 11 z 34

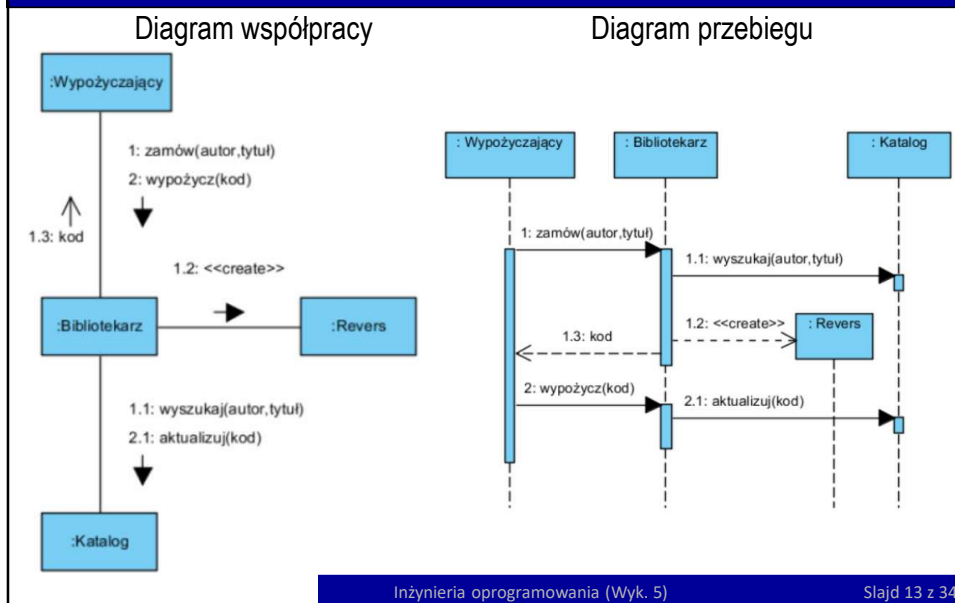
Diagramy współpracy - przykład



Inżynieria oprogramowania (Wyk. 5)

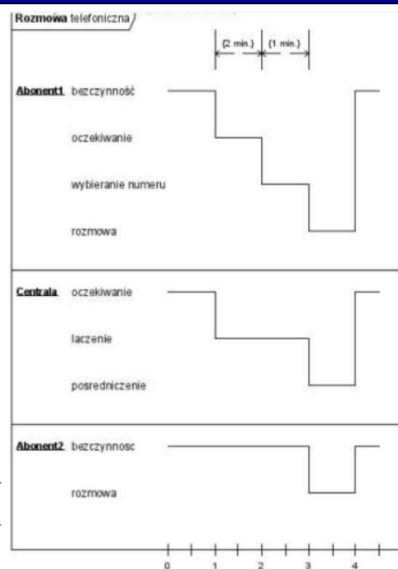
Slajd 12 z 34

Równoważność znaczeniowa (przykład)



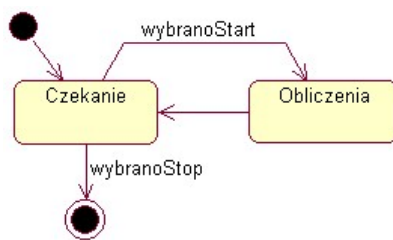
Diagramy czasowe (UML 2.X)

- Modelowanie wielu przebiegów czasowych:
- **Linia stanu** - (ang. state lifeline) wskazuje stan obiektu wyrażony za pomocą wartości dyskretnej; odpowiadają im poziomy linie (różne amplitudy)
- **Linia wartości** – (ang. value lifeline) wskazuje wartość jaką przyjmuje rozpatrywany parametr obiektu (wartość określona tylko za pomocą tekstu)



Maszyny stanowe - podstawowe pojęcia

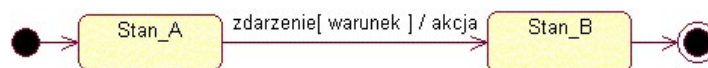
- **Maszyna stanowa** - określa ciąg stanów przyjmowanych przez obiekt w odpowiedzi na zdarzenia zachodzące w czasie jego życia, a także reakcje na te zdarzenia; bardzo przydatne, gdy bieżące zachowanie obiektu zależy od jego przeszłości
- **Zdarzenie** - specyfikacja zjawiska, które zachodzi w czasie i przestrzeni; jest bodźcem, który może uruchomić przejście pomiędzy stanami
- **Stan** - okoliczność lub sytuacja, w jakiej się obiekt znajduje, kiedy spełnia jakiś warunek, wykonuje jakąś czynność lub czeka na jakieś zdarzenie; zwykle obiekt pozostaje w pewnym stanie przez skończony czas
- **Przejście** - związek między dwoma stanami; wskazuje, że obiekt znajdujący się w pierwszym stanie wykona pewne akcje i przejdzie do drugiego stanu, o ile zajdzie określone zdarzenie i będą spełnione określone warunki
- **Akcja** - wykonywalna niepodzielna procedura obliczeniowa prowadząca do zmiany stanu systemu lub do przekazania wartości



Inżynieria oprogramowania (Wyk. 5)

Slajd 15 z 34

Składniki przejść



zdarzenie_uruchamiające [warunek_dozoru] / akcja

- **Stan źródłowy** - gdy obiekt jest w stanie źródłowym i nastąpi zdarzenie uruchamiające, a warunek dozoru (o ile istnieje) jest spełniony, przejście może być uruchomione; do czasu aż przejście nie zostanie uruchomione, obiekt pozostaje w stanie źródłowym
- **Akcja** - wykonywalna niepodzielna procedura obliczeniowa, która może mieć bezpośredni wpływ na obiekt będący właścicielem maszyny stanowej i pośredni wpływ na inne obiekty znajdujące się w jego zasięgu
 - może to być wywołanie operacji (obiektu lub innych dostępnych obiektów), utworzenie lub zniszczenie obiektu oraz wysłanie sygnału do obiektu
 - nie może być przerwana przez zdarzenie - zawsze jest wykonywana w całości
- **Stan docelowy** - stan obiektu po zakończeniu przejścia

Inżynieria oprogramowania (Wyk. 5)

Slajd 16 z 34

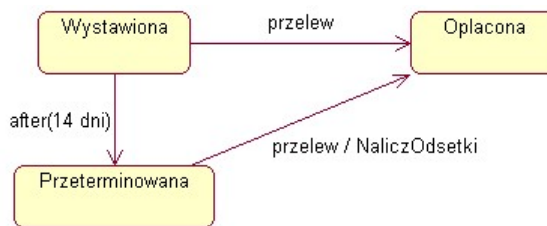
Zdarzenia uruchamiające

Zdarzenie uruchamiające (ang. *event trigger*)- zdarzenie oznaczające, że przejście może nastąpić (sygnał, wywołane operacji, upływ czasu i zmiana stanu)

- sygnał lub wywołanie może mieć parametry, których wartości są dostępne w ramach przejścia
- przejścia automatyczne (bez zdarzenia uruchamiającego) - przejścia (tzw. *zakończeniowe*) uruchamiane są natychmiast po zakończeniu czynności w stanie źródłowym

• zdarzenia czasowe:

- **after** (okres_czasu)
np. after(3 miesiące),
after(1 godzina)
- **when** (moment_czasu)
np. when(31.12),
when (godzina 22.00)

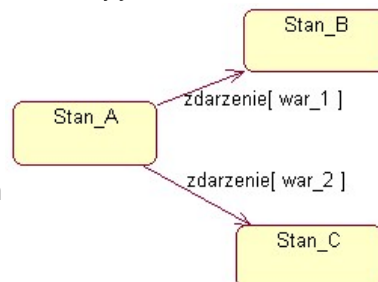


Faktura

Składniki przejść (2)

Warunek dozoru (ang. *guard condition*) - wyrażenie logiczne, którego wartość jest wyznaczana w chwili otrzymania zdarzenia uruchamiającego i spełnienie go warunkuje dokonanie przejścia

- obrazowany w postaci wyrażenia logicznego w nawiasach kwadratowych tuż za zdarzeniem uruchamiającym
- można określić wiele przejść z tego samego źródła i z tym samym zdarzeniem uruchamiającym o ile warunki nadzoru się nie nakładają
- zdarzenie uruchamiające jest ignorowane, gdy żadne przejście przez nie inicjowane nie może dość do skutku
- warunek dozoru może występować bez zdarzenia uruchamiającego => zmiana wartości wyrażenia skutkuje umożliwieniem przejścia



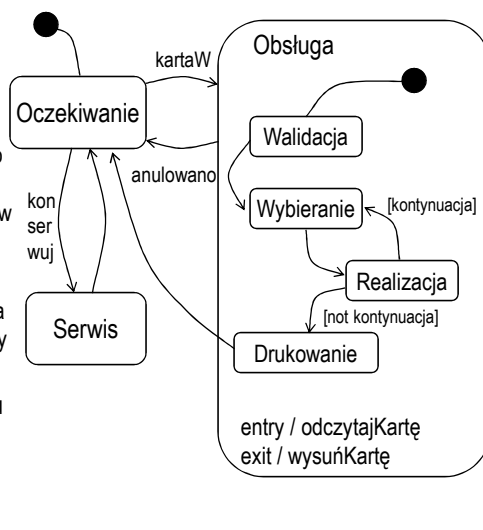
Złożone elementy stanów i przejść

- **Akcje wejściowe (entry) i wyjściowe (exit)** - wykonywane zawsze, gdy przyjmowany / opuszczany jest stan; w zasadzie nie mogą mieć argumentów ani warunków dozoru
- **Przejścia wewnętrzne** - do zdarzeń obsługiwanych bez wyjścia ze stanu; ilekroć zachodzi zdarzenie to skojarzona z nim akcja jest realizowana bez opuszczania stanu (nie są wykonywane akcje wyjściowe i wejściowe)
- **Czynności** - będąc w jakimś stanie obiekt może realizować pewne zadania aż do chwili zajścia zdarzenia; słowo kluczowe "do" służy do wskazania prac wykonywanych w danym stanie od chwili zakończenia akcji wejściowej (może to być uruchomienie innej maszyny stanowej lub ciąg akcji, rozdzielonych średnikami)
- **Zdarzenia odroczone** - lista zdarzeń, których zajście w stanie jest odroczone do czasu, aż stan, w którym nie są odroczone, się uaktywni - dopiero wtedy te zdarzenia powodują przejścia, jakby właśnie zasły; zdarzenia takie oznaczają się specjalnym rodzajem akcji defer

Śledzenie
 entry / ustawTryb(szuka)
 exit / ustawTryb(nieSzuka)
 nowyCel / celownik.Rozpoznaj()
 do / śledźCel
 autoTest / defer

Podstany sekwencyjne

- Podstany sekwencyjne dzielą przestrzeń stanów złożonego na stany rozłączne
- Przejście od źródła znajdującego się na zewnątrz może prowadzić do stanu złożonego (wtedy musi być określony stan początkowy) lub bezpośrednio do jednego z jego podstanów
- Niezależnie od przyjętego rozwiązania akcja wejściowa stanu złożonego jest realizowana
- Zagnieżdżona sekwencyjna maszyna stanowa może mieć co najwyżej jeden stan początkowy i jeden stan końcowy
- Aby zapamiętać aktualny podstan obiektu po opuszczeniu używamy **stanów wznowienia** (płytkie i głębokie)

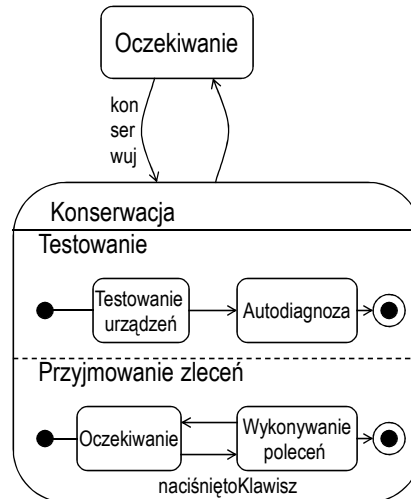


(H)

(H*)

Podstany współbieżne

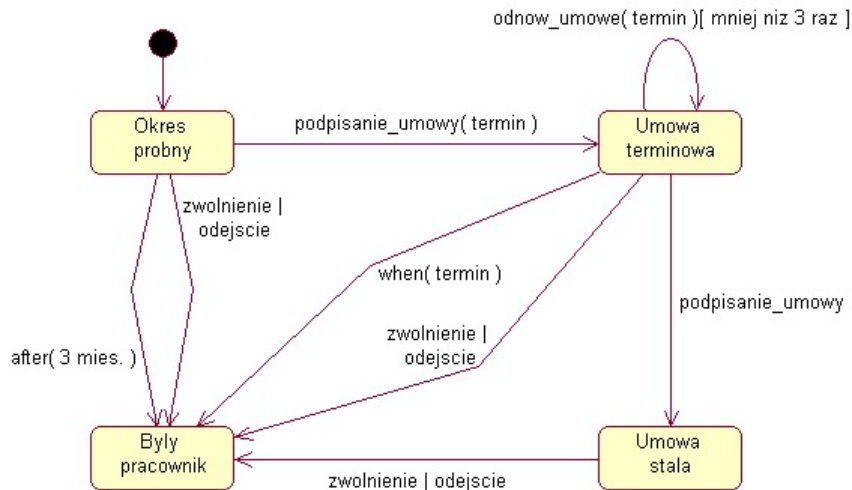
- Umożliwiają tworzenie dwu lub więcej maszyn stanowych, działających równolegle w ramach jednego obiektu
- Czynności podstanów współbieżnych przebiegają równolegle i ostatecznie każda z nich osiąga swój stan końcowy; jeżeli jeden dotrze do swego stanu końca wcześniej to czeka na pozostałe i dopiero wtedy następuje połączenie przepływu sterowania
- Zagnieżdżona współbieżna maszyna stanowa nie ma jako całość ani stanu początkowego ani końcowego; natomiast mogą je mieć składowe podstany sekwencyjne



Diagramy stanów

- Przedstawiają maszyny stanowe z uwypukleniem przepływu sterowania między stanami; pokazują jak zachowanie obiektów zależy od kolejności zachodzących zdarzeń
- Zawierają:
 - stany zwykłe (proste) i złożone
 - przejścia ze zdarzeniami i akcjami
- Najczęściej wykorzystywane do modelowania obiektów reaktywnych (sterowanych zdarzeniami - ang. *event-driven*)
 - zachowanie obiektów reaktywnych jest najlepiej charakteryzowane przez ciąg odpowiedzi na zdarzenia wywołane w jego otoczeniu, przy czym obiekt taki jest zwykle beczynny do chwili zajścia zdarzenia
 - reakcja na konkretne zdarzenie najczęściej zależy od wcześniejszych zdarzeń
 - nacisk kładziony jest na stany stabilne, zdarzenia uruchamiające przejścia i akcje wykonywane po każdej zmianie stanu

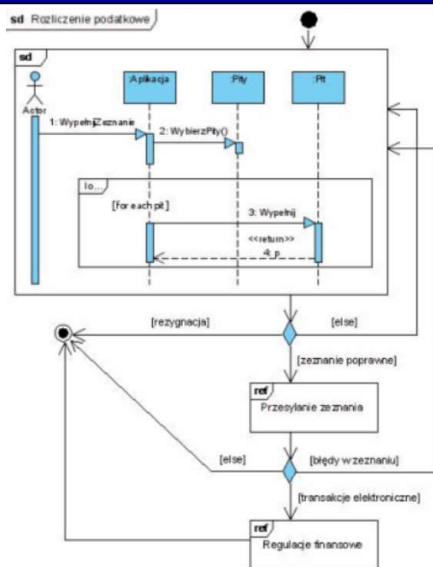
Diagram stanów - przykład (ścieżka zatrudnienia pracownika)



Inżynieria oprogramowania (Wyk. 5)

Slajd 23 z 34

Diagram przeglądu interakcji (UML 2.X)



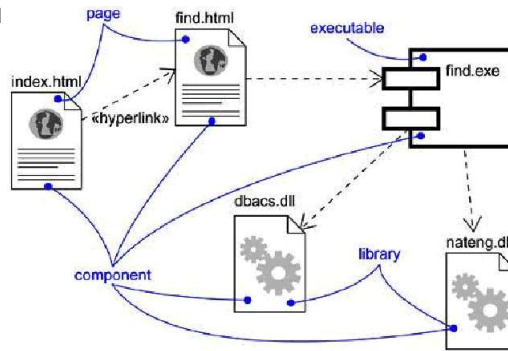
- Połączenie diagramu czynności i diagramu przebiegu
- Diagram ma strukturę wziętą z diagramu czynności (zgrabnie przedstawione warunki), ale zamiast elementów czynnościowych występują interakcje w postaci diagramów przebiegu
- Zastosowanie:
 - modelowanie systemu na wysokim poziomie abstrakcji;
 - przedstawienie połączeń pomiędzy scenariuszami

Inżynieria oprogramowania (Wyk. 5)

Slajd 24 z 34

Diagramy komponentów

- Pierwszy z dwóch rodzajów diagramów przedstawiających fizyczne aspekty systemów obiektowych
- Przedstawiają uporządkowanie komponentów i zależności między nimi; służy do modelowania statycznych aspektów perspektywy implementacyjnej systemu, w których bierze się przede wszystkim pod uwagę zarządzanie konfiguracją poszczególnych części systemu
- Zawierają:
 - komponenty (lub ich egzemplarze)
 - interfejsy
 - zależności, uogólnienia, powiązania i realizacje
- Podobne do diagramu klas (komponenty w roli klas)



Inżynieria oprogramowania (Wyk. 5)

Slajd 25 z 42

Komponenty - wprowadzenie

- Komponent to fizyczna, wymienna część systemu, która wykorzystuje i realizuje pewien zbiór interfejsów; na diagramie przedstawiany jako prostokąt z bolcami
- Służą do modelowania elementów fizycznych, które mogą być umieszczane na węzłach; są to m. in. programy wykonalne, biblioteki, tabele, pliki i dokumenty
- Komponent to fizyczne opakowanie bytów logicznych takich jak klasy, interfejsy i kooperacje
- Nazwę komponentu podaje się zwykle w formie krótkiego rzeczownika lub wyrażenia rzeczownikowego, pochodzącego ze słownictwa implementacji; nazwa komponentu obejmuje też rozszerzenia pliku, zależne od systemu operacyjnego (np. .java czy .dll)

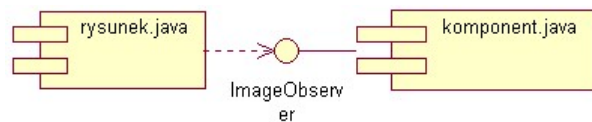


Inżynieria oprogramowania (Wyk. 5)

Slajd 26 z 42

Komponenty i interfejsy

- Interfejs to zestaw operacji, które wyznaczają usługi oferowane przez klasę lub komponent
- Umożliwia to podzielenie fizycznej implementacji na części: określone są interfejsy, które reprezentują podstawowe szwy w systemie; dostarczane są komponenty realizujące te interfejsy oraz komponenty, które korzystają z usług przez te interfejsy
- Interfejs realizowany przez komponent nazywamy **eksportowanym** (komponent udostępnia usługi innym poprzez ten interfejs); komponent może mieć wiele interfejsów eksportowanych

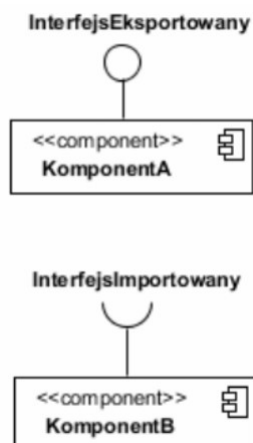


Inżynieria oprogramowania (Wyk. 5)

Slajd 27 z 42

Komponenty i interfejsy (2)

- Interfejs z którego komponent korzysta nosi nazwę **importowanego** (komponent jest zgodny z tym interfejsem i na jego podstawie buduje swoje usługi); komponent może korzystać z wielu interfejsów importowanych
- Jeden komponent może zarówno importować jak i eksportować interfejsy
- Istnienie interfejsu pomiędzy komponentami rozbija ich bezpośrednią zależność (komponent korzystający z danego interfejsu będzie działał poprawnie niezależnie od tego, jaki komponent realizuje ten interfejs)



Inżynieria oprogramowania (Wyk. 5)

Slajd 28 z 42

Zastępstwo na poziomie kodu binarnego

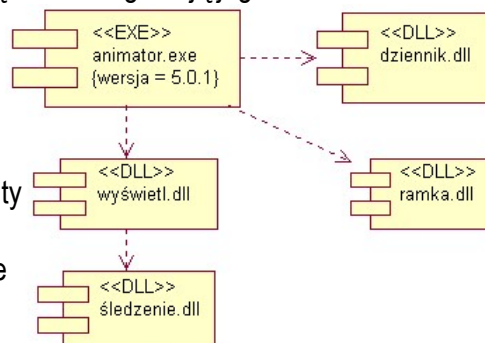
- Wszystkie udogodnienia komponentowe mają za zadanie umożliwienie złożenie systemu z części wymiennych na poziomie kodu binarnego; oznacza to, że można zbudować system z komponentów a następnie rozwijać go poprzez dodawanie nowych komponentów i wymianę starych
- Komponent - **fizyczna i wymienna część systemu**, która wykorzystuje i realizuje pewien zbiór interfejsów
 - fizyczny - istnieje w świecie bitów, a nie pojęć
 - wymienny - może zostać zastąpiony przez inny korzystający z tych samych interfejsów
 - część systemu - rzadko występuje samodzielnie; współpracuje z innymi komponentami i w ten sposób wplata się w otoczenie architektoniczne lub technologiczne; jest fizycznie i logicznie spójny - strukturalna lub czynnościowa porcja systemu; potencjalnie wielokrotnie wykorzystywany w wielu systemach; reprezentuje podstawowy blok konstrukcyjny

Inżynieria oprogramowania (Wyk. 5)

Slajd 29 z 42

Modelowanie plików wykonywalnych i bibliotek

- Jeżeli system składa się z wielu plików wykonywalnych i licznych bibliotek obiektowych to przy użyciu komponentów można zobrazować decyzje projektowe dotyczące systemu fizycznego
- Rola tego procesu jest jeszcze większa, gdy chce się panować nad wieloma wersjami systemu i zarządzać konfiguracją jego składników
- Zobrazowanie zależności między komponentami jest w istocie skróconą formą przedstawienia prawdziwego związku (komponenty zwykle są niezależne, jedynie importują interfejsy eksportowane przez inny komponent)

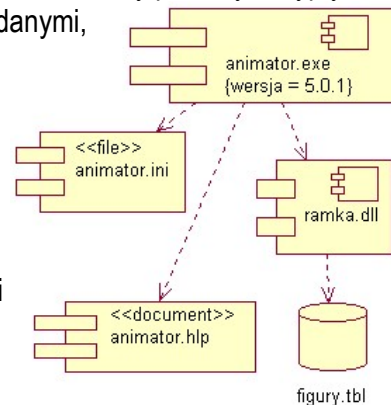


Inżynieria oprogramowania (Wyk. 5)

Slajd 30 z 42

Modelowanie tabel, plików i dokumentów

- Istnieją inne rodzaje (niż programy wykonywalne i biblioteki) pomocniczych komponentów, które są niezbędne w działającym systemie
- Składnikami implementacji mogą być np. dokumenty pomocy, skrypty oraz pliki dzienników, inicjalizacyjne, z danymi, instalacyjne i z procedurami kasowania
- Modelowanie takich komponentów jest istotną częścią procesu zarządzania konfiguracją systemu
- Najczęściej występującymi związkami pomiędzy komponentami pomocniczymi i właściwymi są zależności wskazujące wpływ potencjalnych zmian jednego składnika na drugi

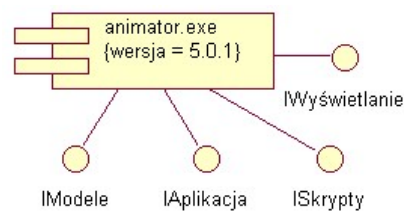


Inżynieria oprogramowania (Wyk. 5)

Slajd 31 z 42

Modelowanie interfejsu programowego (API)

- Programista budujący system z gotowych komponentów musi znać ich interfejsy programowe, umożliwiające łączenie ich ze sobą.
- Interfejsy te reprezentują szwy systemu, który modelujemy za pomocą interfejsów i komponentów
- Operacje wchodzące w skład niebanalnego API, są zwykle bardzo liczne; ich listę warto zapamiętać wewnątrz modelu i używać interfejsów jako wygodnych uchwytów, za pomocą których można mieć dostęp do tych zbiorów operacji
- Należy obrazować jedynie te elementy interfejsu, które są istotne w danym otoczeniu



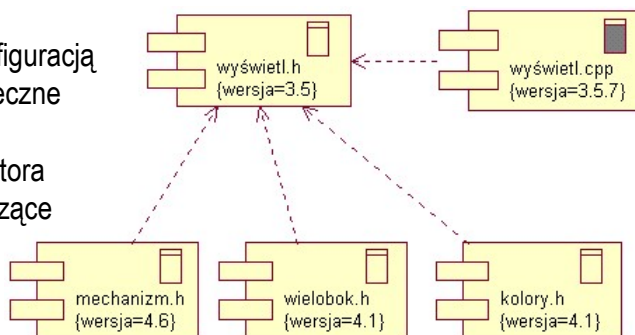
Inżynieria oprogramowania (Wyk. 5)

Slajd 32 z 42

Modelowanie kodu źródłowego

- Graficzne modelowanie kodu źródłowego jest szczególnie użyteczne do obrazowania zależności kompilacyjnych między plikami
- Ułatwia to panowanie nad podziałem i łączeniem grup tych plików, gdy konieczne jest np. rozwidlanie i scalenie ścieżek procesu wytwórczego

- Do zarządzania konfiguracją i kontroli wersji użyteczne są metki określające np. numer wersji, autora lub informacje dotyczące pobrania



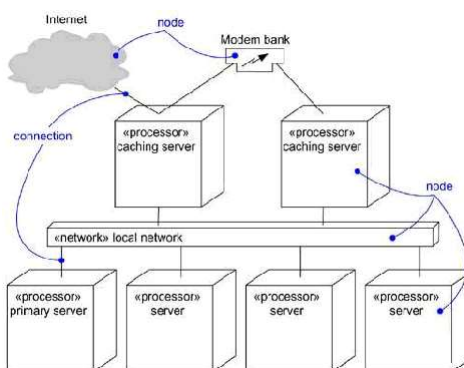
Inżynieria oprogramowania (Wyk. 5)

Slajd 33 z 42

Diagramy wdrożenia

- Drugi z dwóch rodzajów diagramów przedstawiających fizyczne aspekty systemów obiektowych
- Obrazuje konfigurację węzłów działających w czasie wykonania i zainstalowane na nich komponenty
- Bierze pod uwagę przede wszystkim rozproszenie, dostarczenie i instalację części systemu
- Zawiera:
 - węzły
 - zależności i powiązania

- Komponenty mogą występować na diagramie wdrożenia o ile są przypisane do jakiegoś węzła



Inżynieria oprogramowania (Wyk. 5)

Slajd 34 z 42

Wprowadzenie - węzły

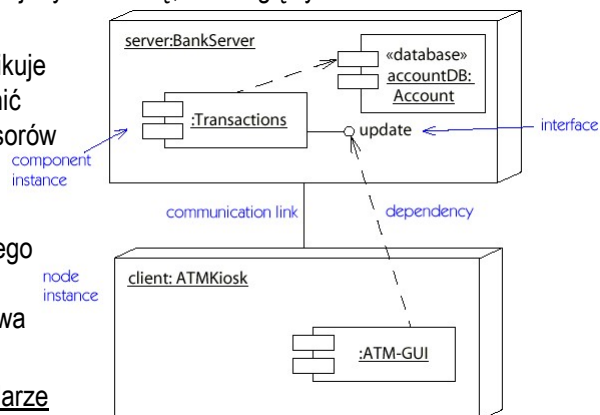
- Wszystkie komponenty systemu informatycznego są wdrażane na sprzęcie komputerowym - niezależnie od tego czy napisano je od nowa, czy wykorzystano od nowa => SI ze swej natury obejmuje zarówno oprogramowanie jak i sprzęt
- Opracowując architekturę SI należy rozważać zarówno wymiar logiczny (klasy, interfejsy, ...) jak i fizyczny (komponenty reprezentujące fizyczne opakowanie bytów logicznych oraz węzły reprezentujące sprzęt na którym te komponenty są posadowiane)
- **Węzeł** to fizyczny składnik działającego systemu; reprezentuje zasoby obliczeniowe; ma zwykle pewną ilość pamięci i zdolność przetwarzania
- Węzłów używa się do modelowania układu sprzętu komputerowego, na którym działa system; zwykle reprezentują procesory lub urządzenia, na których wdrażane są komponenty

Inżynieria oprogramowania (Wyk. 5)

Slajd 35 z 42

Wdrożenie

- Węzeł przedstawiany jest w postaci sześciangu z nazwą (prostą lub ścieżkową); zwykle symbol węzła zawiera jedynie nazwę, ale mogą być metki i dodatkowe sekcje
- Stosując stereotypy modyfikuje się ten symbol, aby wyróżnić specyficzne rodzaje procesorów i urządzeń
- W praktyce nazwę podaje się na ogół w formie krótkiego rzeczownika lub wyrażenia pochodzącego ze słownictwa implementacji
- Węzły mogą mieć egzemplarze
- Zbiór obiektów i komponentów przypisanych węzłowi nazywamy *jednostką rozproszenia*

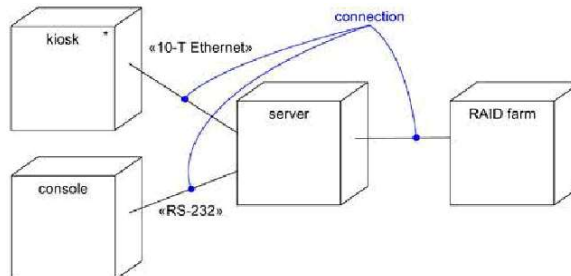


Inżynieria oprogramowania (Wyk. 5)

Slajd 36 z 42

Połączenia

- Najczęściej występującym związkiem pomiędzy węzłami jest **powiązanie**, które w tym przypadku oznacza połączenie fizyczne (np. sieć Ethernet, łącze szeregowe lub wspólna szyna)
- Powiązania można użyć też do modelowania połączeń pośrednich (np. komunikacja satelitarna między odległymi maszynami)
- W przypadku powiązania węzłów (analogicznie jak dla powiązania klas) mogą być wykorzystane role, liczebność i ograniczenia



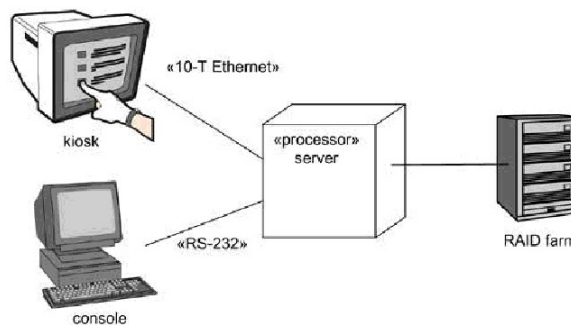
- Aby rozróżnić rodzaje połączeń (np. sieć Ethernet od łącza szeregowego) warto używać stereotypów dotyczących powiązań
- Węzły mogą być również połączone związkiem **uogólnienia**, aby pokazać ogólny opis węzła z jego specyficznymi wariantami

Inżynieria oprogramowania (Wyk. 5)

Slajd 37 z 42

Modelowanie procesorów i urządzeń

- Standardowe konfiguracje: wolnostojący komputer, system wbudowany, system klient-serwer i system rozproszony
- Procesor** to węzeł, który ma zdolność przetwarzania, czyli może realizować działania komponentów
- Urządzenie** to węzeł, który nie ma zdolności przetwarzania (lub nie jest ona modelowana) i zwykle reprezentuje sprzęg ze światem zewnętrznym
- Stereotypy mogą i powinny służyć do specyfikowania nowych typów procesorów i urządzeń (wprowadzenie specjalnych symboli graficznych)

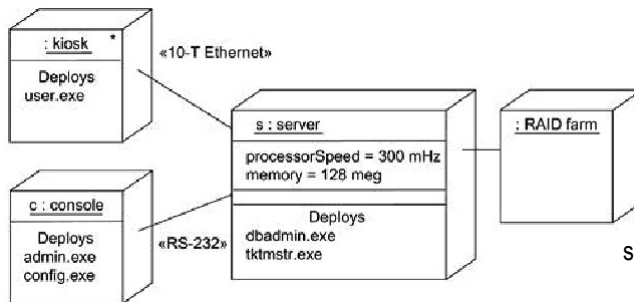


Inżynieria oprogramowania (Wyk. 5)

Slajd 38 z 42

Modelowanie rozproszonych komponentów

- Modelując układ systemu, można zobrazować fizyczne rozproszenie jego komponentów po procesorach i urządzeniach wchodzących w jego skład
- Komponent zwykle jest przypisany do jednego węzła ale może być też umieszczony na kilku (np. specyficzne programy wykonywalne i biblioteki)
- Położenie komponentów jest zwykle obrazowane przez ich wymienienie w dodatkowej sekcji symbolu węzła



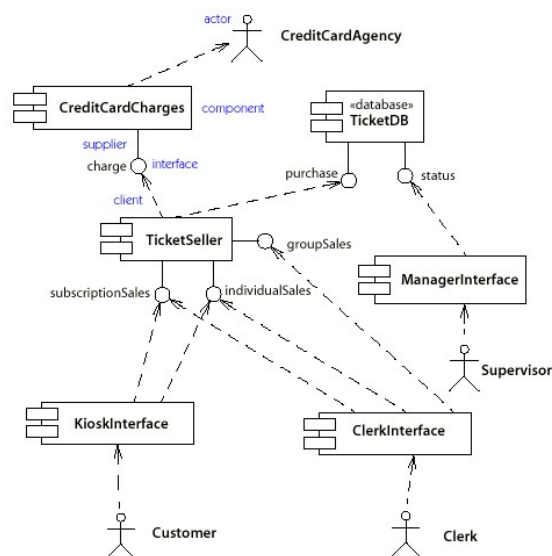
Komponenty nie muszą być przypisane do węzłów na stałe (można modelować dynamiczną migrację komponentów między węzłami, np. w systemach wieloagentowych)

Inżynieria oprogramowania (Wyk. 5)

Slajd 39 z 42

Przykład diagramu komponentów

Box office system
(sprzedaż biletów)

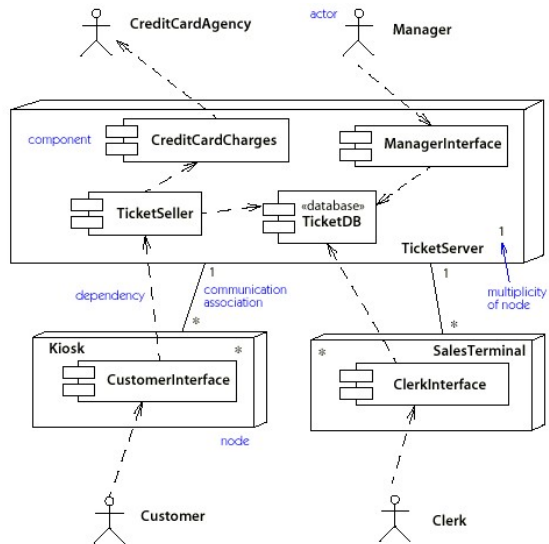


Inżynieria oprogramowania (Wyk. 5)

Slajd 40 z 42

Przykład diagramu wdrożenia - poziom opisowy

Box office system

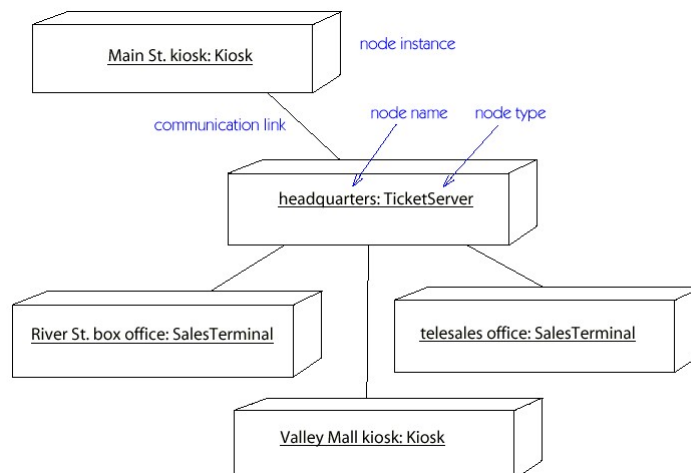


Inżynieria oprogramowania (Wyk. 5)

Slajd 41 z 42

Przykład diagramu wdrożenia - poziom egzemplarzy

Box office system



Inżynieria oprogramowania (Wyk. 5)

Slajd 42 z 42