Podstawy projektowania systemów komputerowych

Dr inż. Gabriel Rojek

Literatura

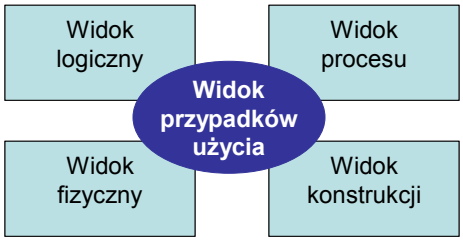
R. Miles, K. Hamilton „**UML 2.0. Wprowadzenie**”, Helion 2007

UML - Unified Modeling Language

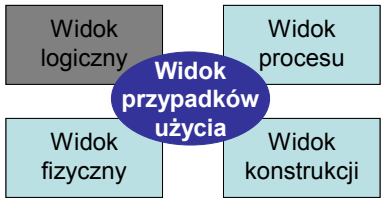
- UML **nie jest** sposobem na analizę i projektowanie systemów komputerowych.
- UML **jest jedynie** językiem modelowania używanym w procesie analizy i projektowania systemów komputerowych.
  - Tworzy się z jego pomocą **model** systemu, inaczej **abstrakcję** (w sensie *skrót, uproszczenie*) systemu.

Widoki modelu

- Jednym ze sposobów rozbijania diagramów UML na perspektywy lub widoki jest **system widoków 4+1 Kruchtena.**

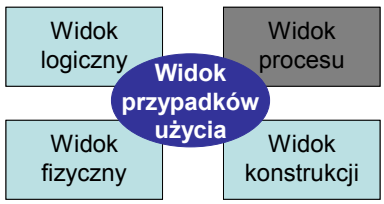


Widok logiczny



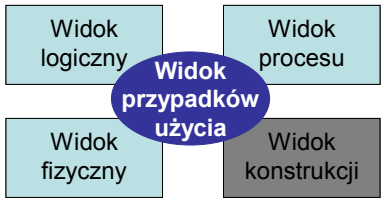
- Używany do modelowania części systemu oraz sposobów, w jaki one ze sobą współdziałają.
- Ten widok zazwyczaj tworzą diagramy:
  - Klas
  - Obiektów,
  - Maszyny stanowej,
  - Interakcji.

Widok procesu



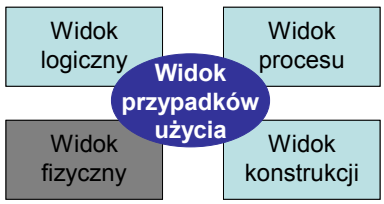
- Opisuje procesy w systemie.
- Przydatny przy wizualizacji przypadków, jakie muszą zajść w systemie.
- Zawiera zazwyczaj diagram czynności.

Widok konstrukcji



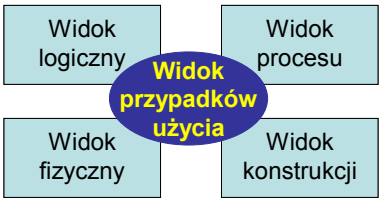
- Opisuje sposób, w jaki części systemu są zorganizowane w moduły oraz komponenty.
- Zawiera zazwyczaj diagramy:
  - Pakietów
  - Komponentów.

Widok fizyczny



- Wyjaśnia w jaki sposób projekt systemu opisany w trzech poprzednich widokach jest powoływany do życia w postaci zestawu rzeczywistych obiektów.
- Rzeczywiste wdrożenie systemu.
- Zawiera zazwyczaj diagramy wdrożenia.

Widok przypadków użycia



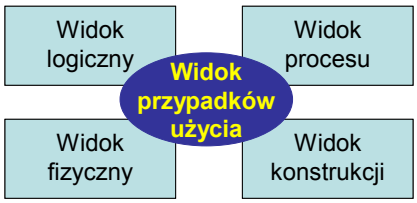
- Widok opisuje funkcjonalność modelowanego systemu **z perspektywy zewnętrznej**.
- Prezentuje przeznaczenie systemu.
- Wszystkie inne widoki bazują na widoku przypadków użycia (nazywa się przecież **4+1**).
- Zawiera zazwyczaj diagramy przypadków użycia, opisy i diagramy przeglądowe.

Modelowanie wymagań: przypadki użycia

Przypadek użycia

- Jest przypadkiem, w którym dany system jest używany w celu spełniania jednego lub większej liczby wymagań użytkowników.
- Wychwytuje fragment funkcji udostępnianych przez system.
- Określają wymagania funkcjonalne systemu.

Przypadki użycia



- Przypadki użycia dotykają każdej innej części projektu systemu.
- Przypadki użycia nie określają wymagań niefunkcjonalnych systemu.

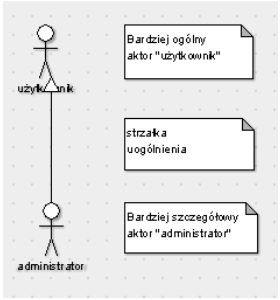
Wychwytywania przypadków użycia

- Analiza opisu systemu z punktu widzenia przyszłego użytkownika.
  - Wymagania konieczne (bez nich system nie będzie funkcjonował),
  - Wymagania opcjonalne (pożądane, mogą być porzucone jako pierwsze w chwili napotkania nieprzewidzianych trudności w realizacji systemu).

Aktorzy

- Aktorzy mogą być:
  - Ludźmi wchodzącymi w interakcję z systemem,
  - Systemami zewnętrznymi,
  - Częściami systemu, które mają wpływ na funkcjonowanie systemu, ale same przez ten system nie mogą być zmieniane (jak np. zegar systemowy).

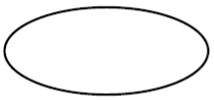
Powiązania pomiędzy aktorami



- Aby pokazać, że administrator może zrobić wszystko to, co zwyczajny użytkownik (a także kilka dodatkowych rzeczy), użyta została strzałka uogólnienia.

Use Case

A use case represents a class of functionality provided by the system as an event flow.



`PurchaseTicket`

A use case consists of:

- Unique name
- Participating actors
- Entry conditions
- Flow of events
- Exit conditions
- Special requirements

Use Case Example

Name: `Purchase ticket`

Participating actor: `Passenger`

Entry condition:

- `Passenger` standing in front of ticket distributor.
- `Passenger` has sufficient money to purchase ticket.

Exit condition:

- `Passenger` has ticket.

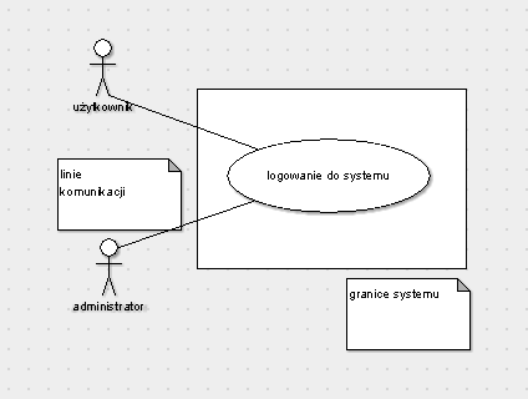
Event flow:

1. `Passenger` selects the number of zones to be traveled.
2. Distributor displays the amount due.
3. `Passenger` inserts money, of at least the amount due.
4. Distributor returns change.
5. Distributor issues ticket.

Anything missing?

Exceptional cases!

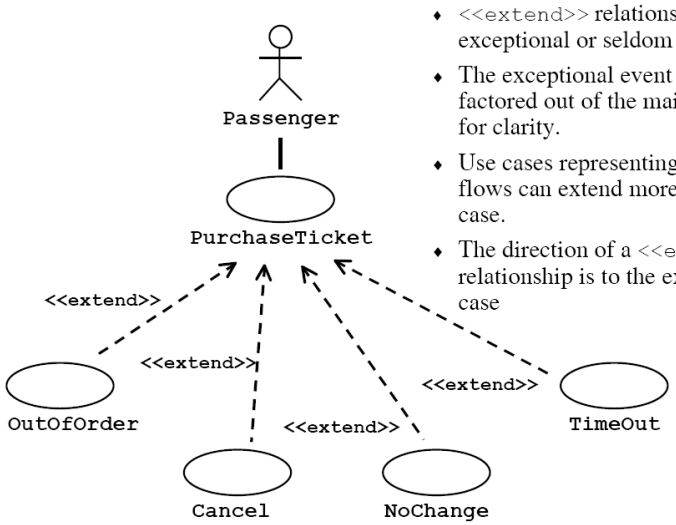
Linie komunikacji, granice systemu



Czy „logowanie do systemu" jest dobrym przypadkiem użycia?

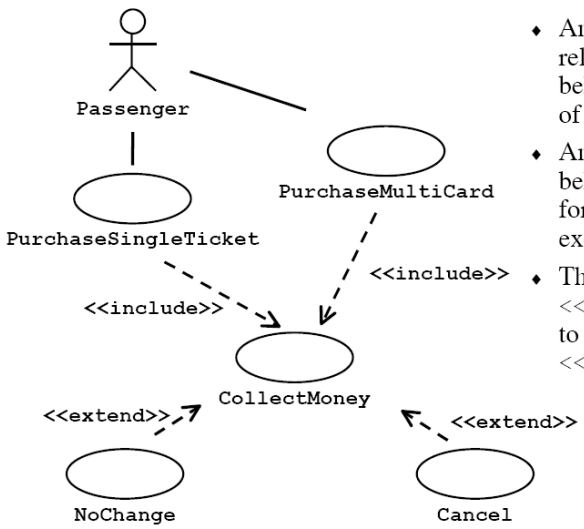
- Linie komunikacji sygnalizują, że aktor uczestniczy w przypadku użycia lub inicjuje go.

The <<extend>> Relationship



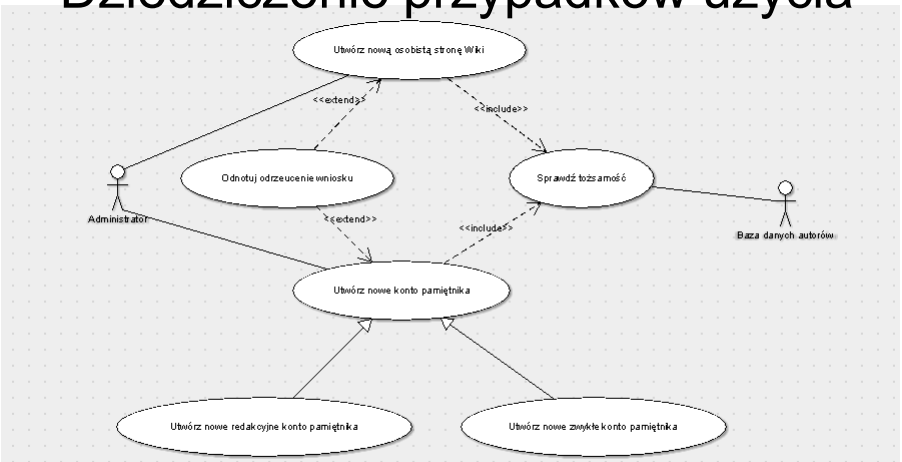
- <<extend>> relationships represent exceptional or seldom invoked cases.
- The exceptional event flows are factored out of the main event flow for clarity.
- Use cases representing exceptional flows can extend more than one use case.
- The direction of a <<extend>> relationship is to the extended use case

The <<include>> Relationship



- An <<include>> relationship represents behavior that is factored out of the use case.
- An <<include>> represents behavior that is factored out for reuse, not because it is an exception.
- The direction of a <<include>> relationship is to the using use case (unlike <<extend>> relationships).

Dziedziczenie przypadków użycia



- Dziedziczenie pozwala na współdzielenie większości zachowania ogólnego przypadku użycia.