# Inżynieria Oprogramowania


# Modelowanie Systemu Informatycznego

prezentacja 3

## Modelowanie wymagań – diagram wymagań i diagram przypadków użycia

wersja 1.0

***dr inż. Paweł Głuchowski***

*Wydział Informatyki i Telekomunikacji, Politechnika Wrocławska*

# Treść prezentacji

1. Diagram wymagań
2. Diagram przypadków użycia
3. Modelowanie wymagań i przypadków użycia
4. Przykłady

# 1

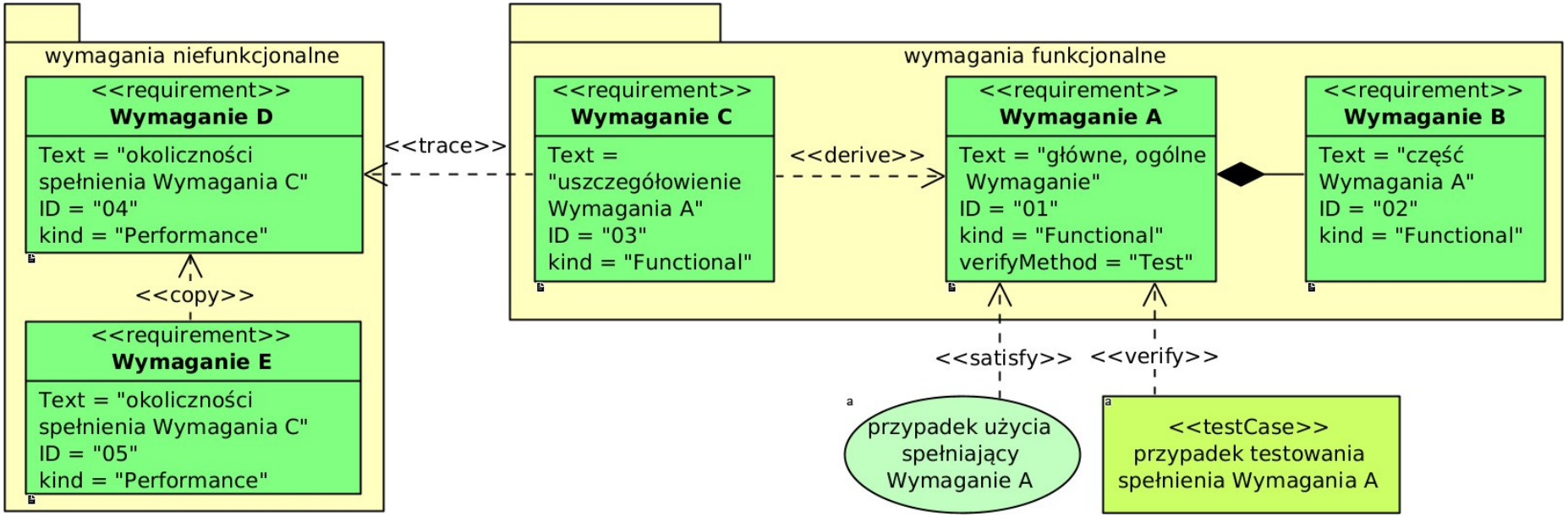
# Diagram wymagań

## Diagram wymagań /requirement diagram/

- Modeluje wymagania funkcjonalne i niefunkcjonalne stawiane systemowi:
  - warunki lub cele do spełnienia lub spełniania,
  - relacje między nimi.
- **Wymaganie** /requirement/ to NIE proces realizacji jakiegoś celu, ale sam cel.
- Opracowany na podstawie słownej specyfikacji wymagań i (czasami) przypadków użycia.
- W języku zrozumiałym przez klienta i użytkownika systemu.

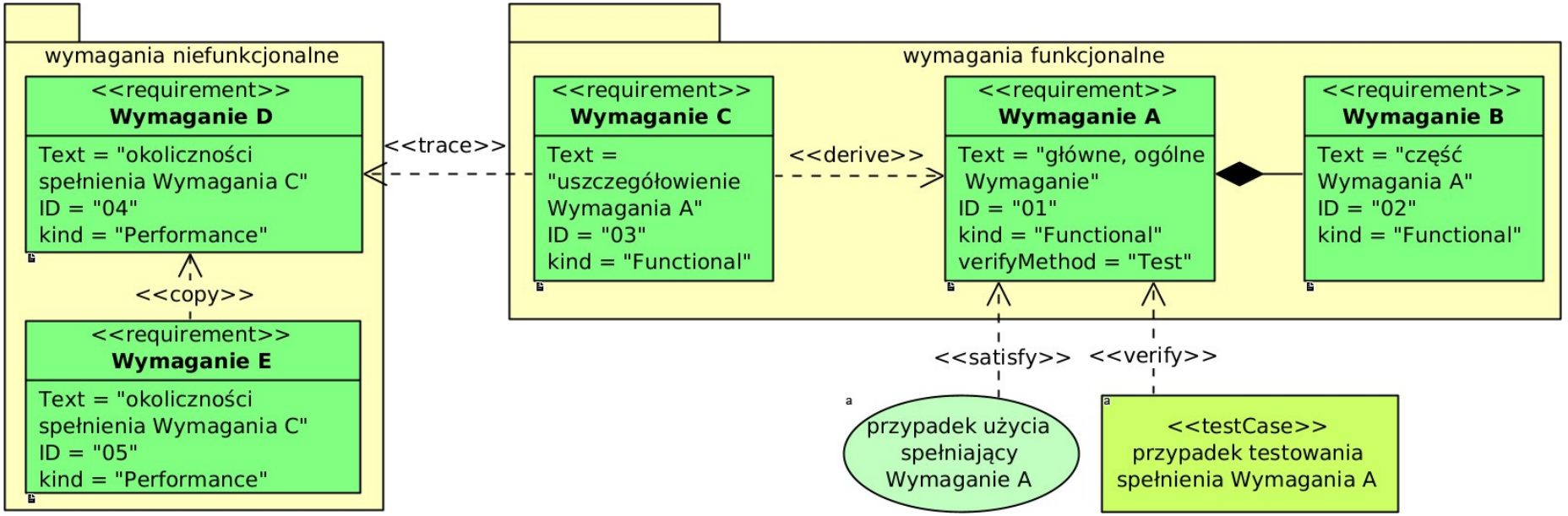
## Wymaganie funkcjonalne

- Opisuje zadanie, które system musi wykonać lub wykonywać
  - co system ma osiągnąć lub w jakim być stanie;
  - co system ma robić, aby zautomatyzować działania jego użytkowników.



## Wymaganie niefunkcjonalne

- Opisuje jakościowe kryteria efektywności zadań systemowych
  - jak lub w jakich warunkach system ma to osiągnąć (przy jakich ograniczeniach sprzętowych, organizacyjnych, prawnych…);
  - jakie zastosować rozwiązania technologiczne m.in. w zakresie: bezpieczeństwa, niezawodności, skalowalności i wydajności, aby spełnić oczekiwania użytkownika/klienta.

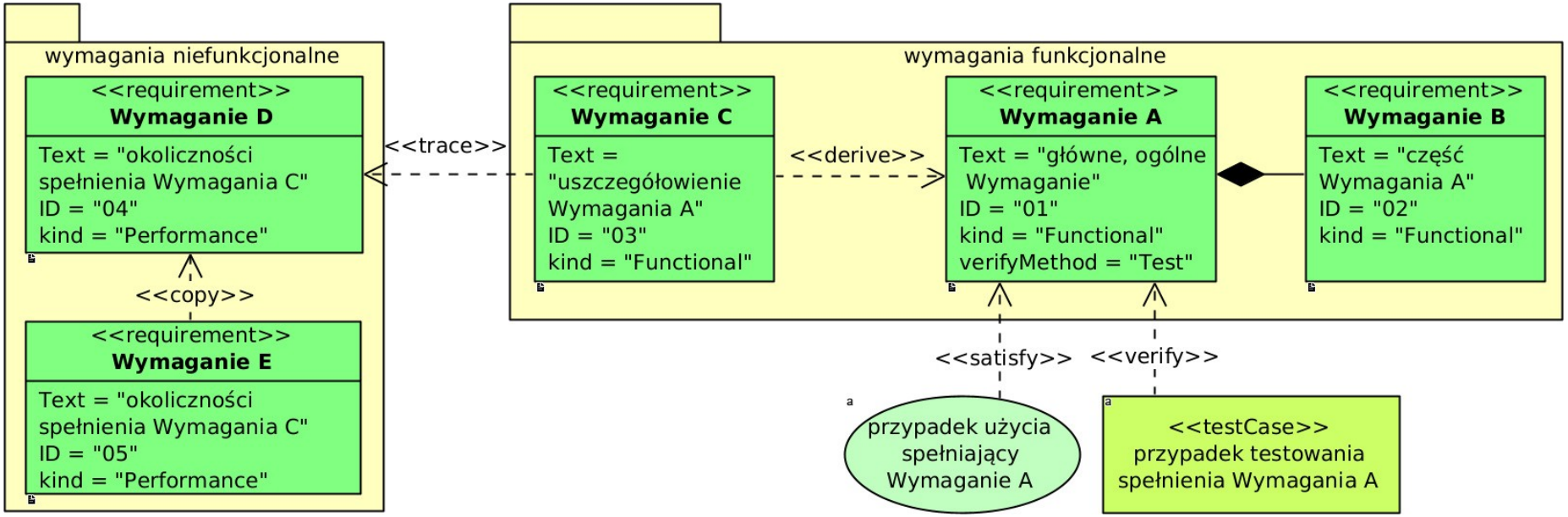


## Formułowanie wymagań systemu

- **Perspektywa obserwatora:**
  - bezosobowo – mówi obserwator systemu,
  - deklaratywnie – stwierdzenie faktu,
  - przykłady:
    - *„Kontroler bezpieczeństwa sprawdza sprawność systemu alarmowego…”*
    - *„Kontroler bezpieczeństwa może sprawdzać sprawność systemu alarmowego…”*
- **Perspektywa użytkownika:**
  - w 1. osobie – mówi użytkownik systemu:
    - analityk/projektant wchodzi w punkt widzenia użytkownika;
  - życzeniowo – stwierdzenie oczekiwania;
  - przykład:
    - *„Jako kontroler bezpieczeństwa chcę móc sprawdzać sprawność systemu alarmowego…”*

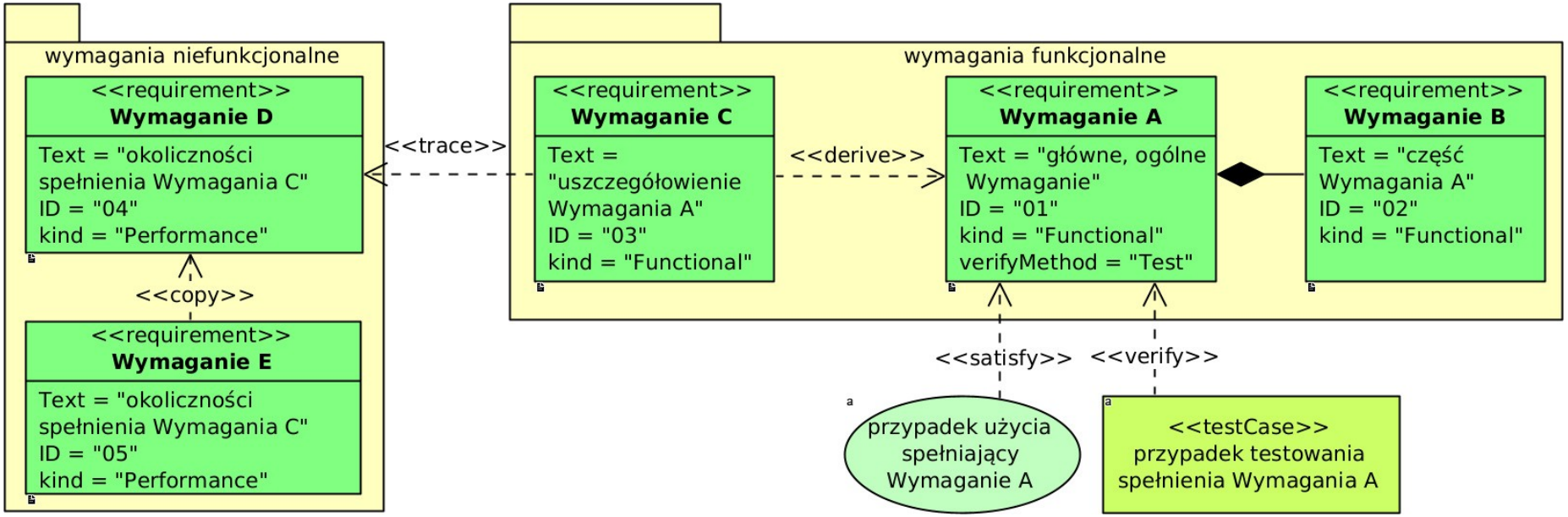
## Przypadek testowania *«testCase»*

- Opisuje sposób sprawdzenia, czy wymaganie zostało spełnione.
- Modeluje: test realizacji przypadku użycia, test systemu (np. funkcjonalny), test metody klasy (jednostkowy) i inne.
- Zakłada stan początkowy przedmiotu testu.
- Zawiera scenariusz testowania w postaci tekstowej (pseudokod) lub graficznej (powiązany z nim diagram czynności).
- Zakłada uzyskanie określonego artefaktu lub stanu końcowego.



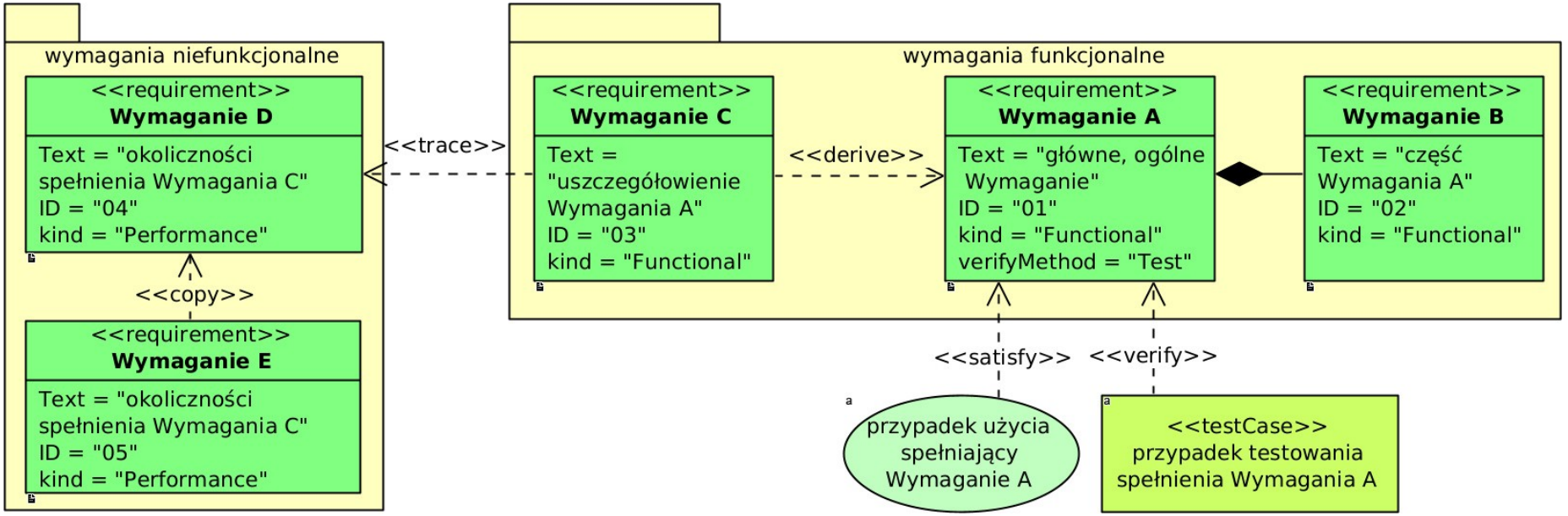
## Relacje powstałe w analizie wymagań

- **Relacja kompozycji** /composition/ – łączy wymagania w związek całość-część (grot ♦ jest przy „całości”).
- **Relacja wyprowadzenia *«derive»*** – wyprowadzenie wskazywanego wymagania ze wskazującego (uszczegóławianego) wymagania.
- **Relacja śladu *«trace»*** – słabsze uzależnienie (uwarunkowanie) wskazującego wymagania przez wskazane wymaganie, będące:
  - ograniczeniem spełnienia wskazującego wymagania,
  - wcześniej spełnionym wymaganiem dla wskazującego wymagania.



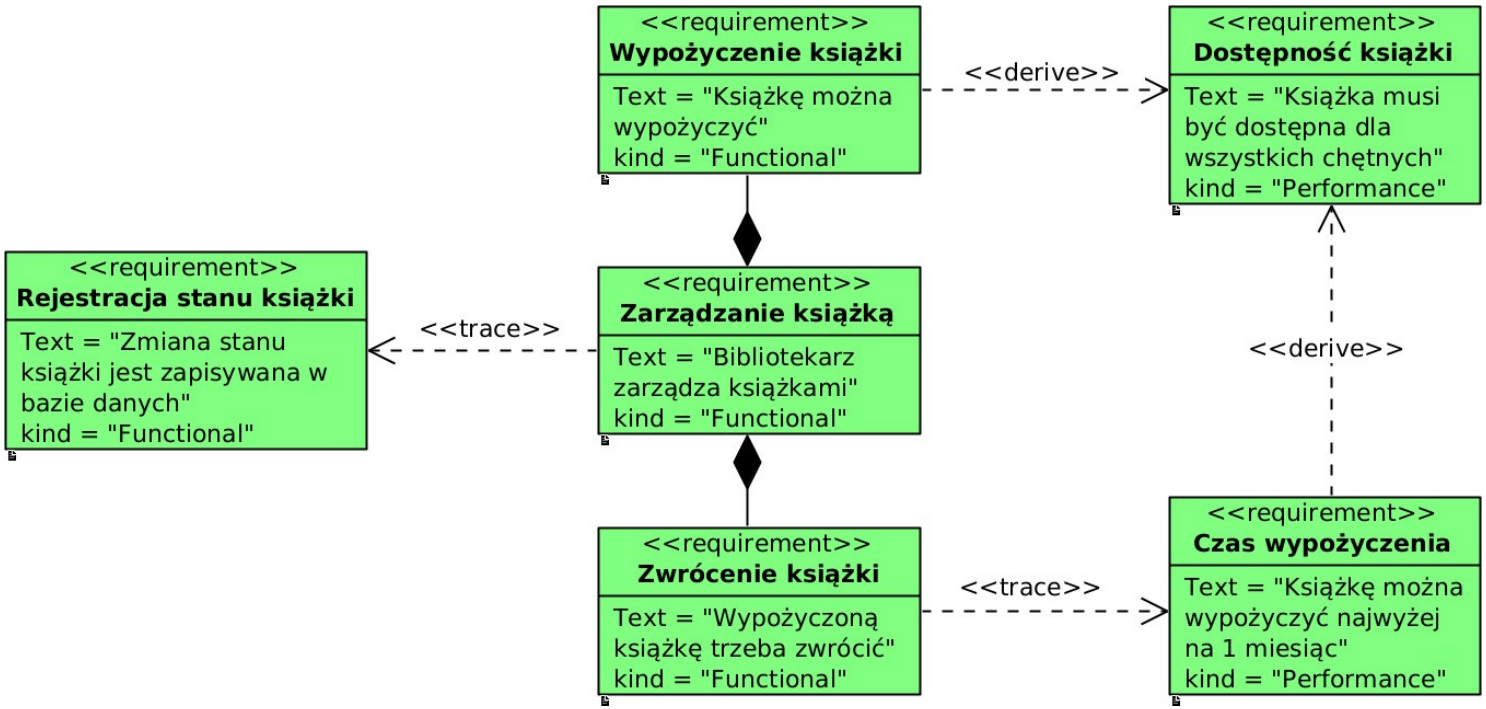
## Relacje powstałe w analizie wymagań

- **Relacja kopii *«copy»*** – utworzenie wskazującego wymagania jako kopii (aliasu) wskazanego wymagania.
  - Kopia ma inną nazwę, ale ten sam opis.
  - Kopię można zmienić tylko przez zmianę oryginału.
- **Relacja spełnienia *«satisfy»*** – spełnienie wskazywanego wymagania przez wskazujący przypadek użycia.
- **Relacja sprawdzenia *«verify»*** – sprawdzenie spełnienia wskazywanego wymagania przez wskazujący przypadek testowania.



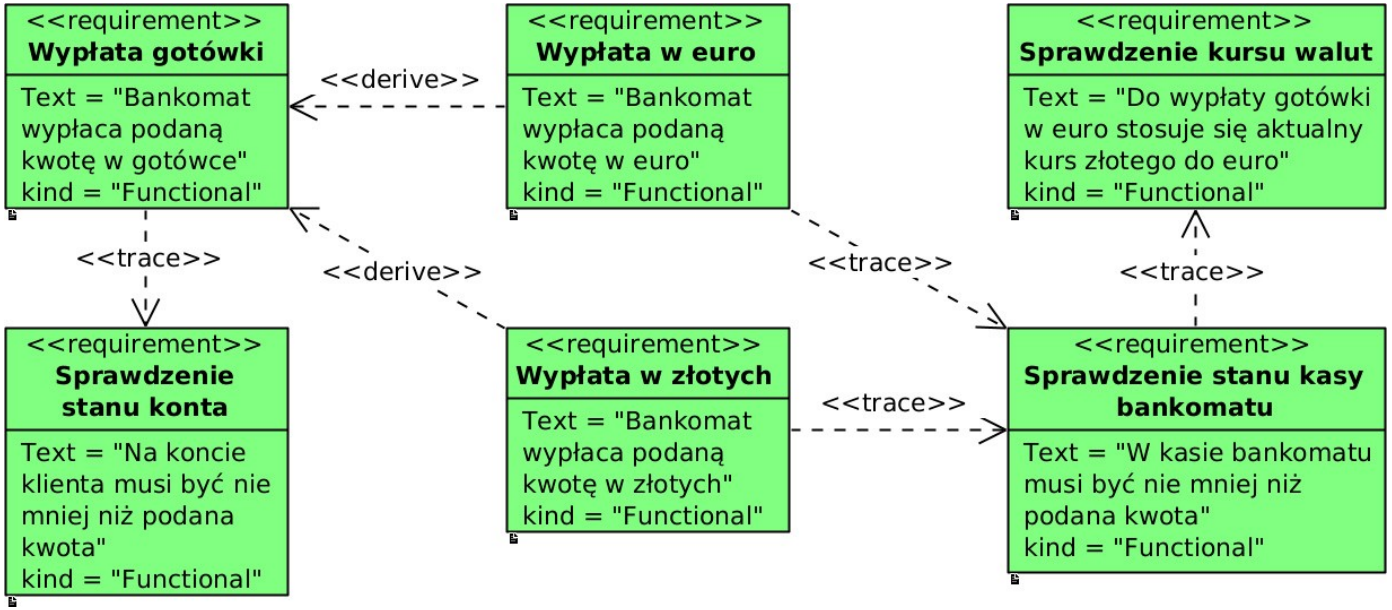
## Przykład relacji kompozycji, śladu i wyprowadzenia

- Zarządzanie książką to (kompozycja) jej wypożyczenie lub zwrócenie.
- Zarządzanie książką prowadzi do (*«trace»*) rejestracji jej stanu.
- Wypożyczenie książki wynika z (*«derive»*) jej dostępności.
- Zwrócenie książki wymaga (*«trace»*) zachowania czasu jej wypożyczenia.
- Czas wypożyczenia książki wynika z («derive») zachowania jej dostępności.



## Przykład relacji śladu i wyprowadzenia

- Wersje wypłaty gotówki (*«derive»*) to: wypłata w euro i wypłata w złotych.
- Wypłata gotówki wymaga (*«trace»*) sprawdzenia stanu konta.
- Wypłata w euro i wypłata w złotych wymaga (*«trace»*) sprawdzenia stanu kasy bankomatu, które wymaga (*«trace»*) sprawdzenia kursu walut.



# 2

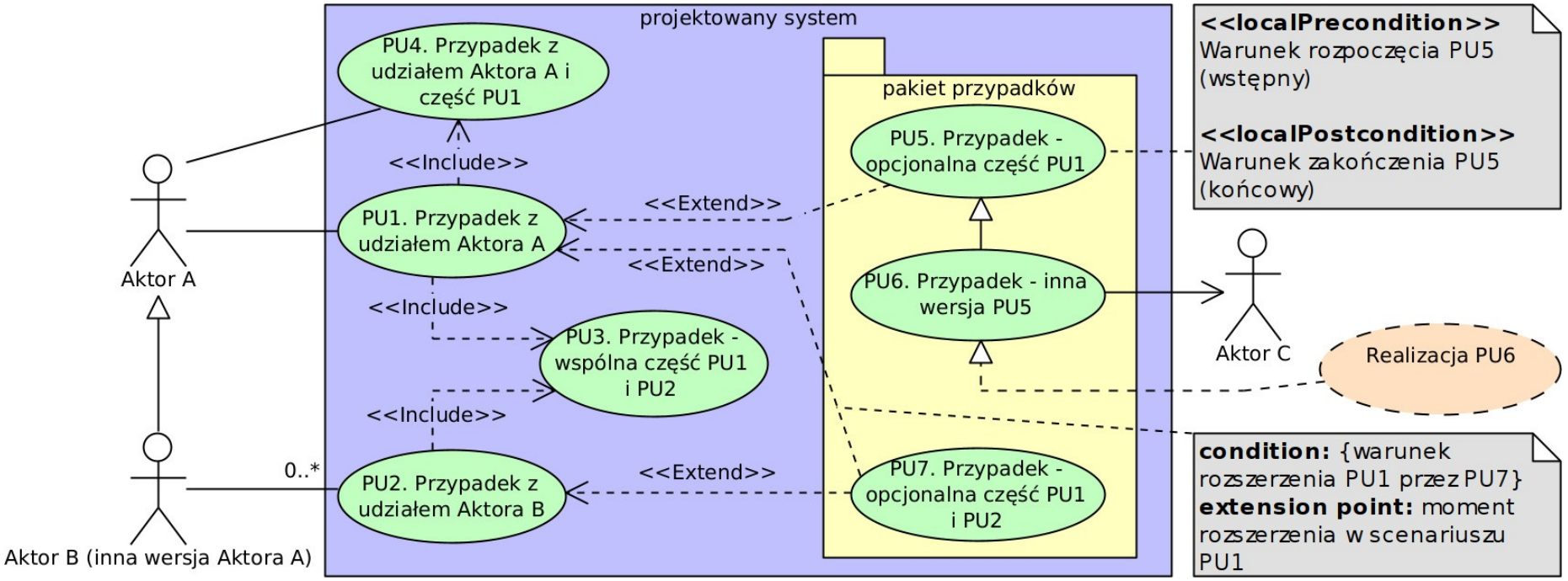
# Diagram przypadków użycia

## Diagram przypadków użycia /use case diagram/

- Modeluje przypadki użycia i aktorów systemu oraz jego podział na podsystemy.
- Modeluje związki między przypadkami użycia i między przypadkami użycia a aktorami.
- NIE modeluje następstwa (kolejności) realizacji przypadków użycia.
- Opracowany na podstawie tekstowego opisu specyfikacji wymagań oraz diagramów wymagań.
- W języku zrozumiałym przez klienta i użytkownika systemu.
- Modeluje zewnętrzną strukturę systemu z jej funkcjonalnością i ogólną koncepcją architektury.

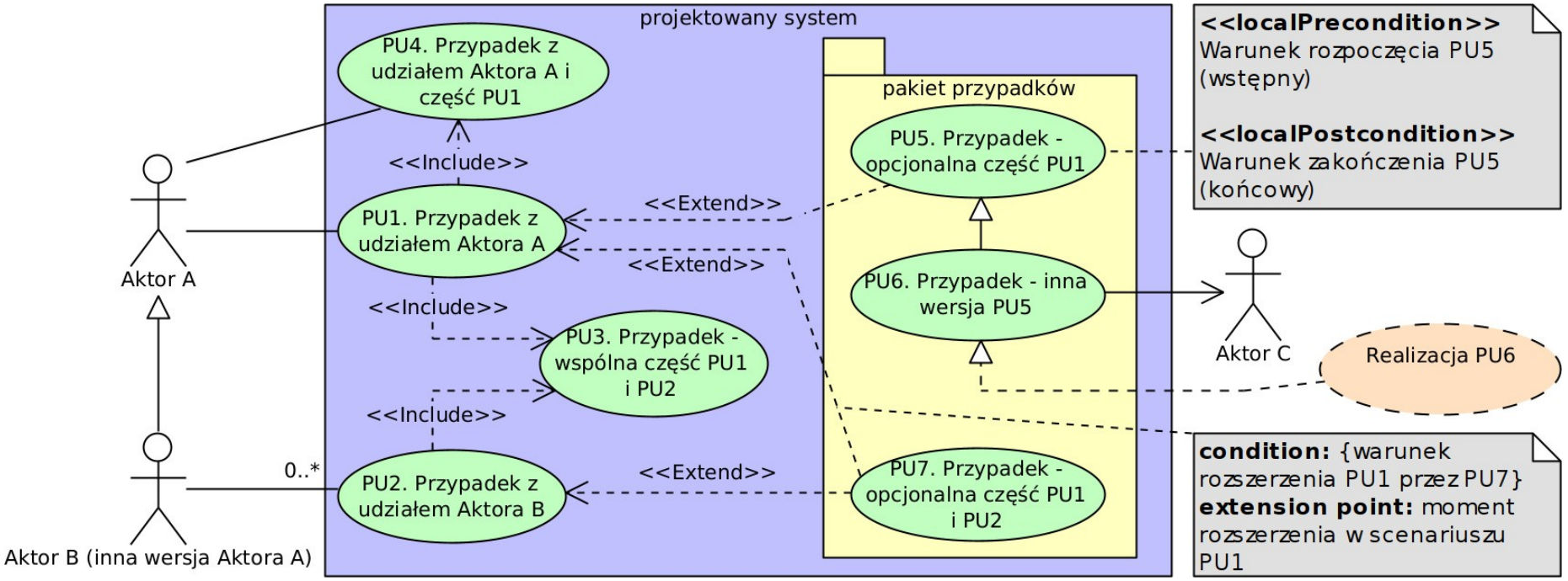
## Przypadek użycia /use case/

- Proces biznesowy spełniający wymagania funkcjonalne.
- Zdefiniowany ogólnie, a NIE jako składowa czynność procesu biznesowego.
- Modeluje działania i oczekiwania aktorów w stosunku do systemu:
  - „funkcje” systemu udostępniane aktorom.
- Przypadki użycia można grupowane w pakiety.



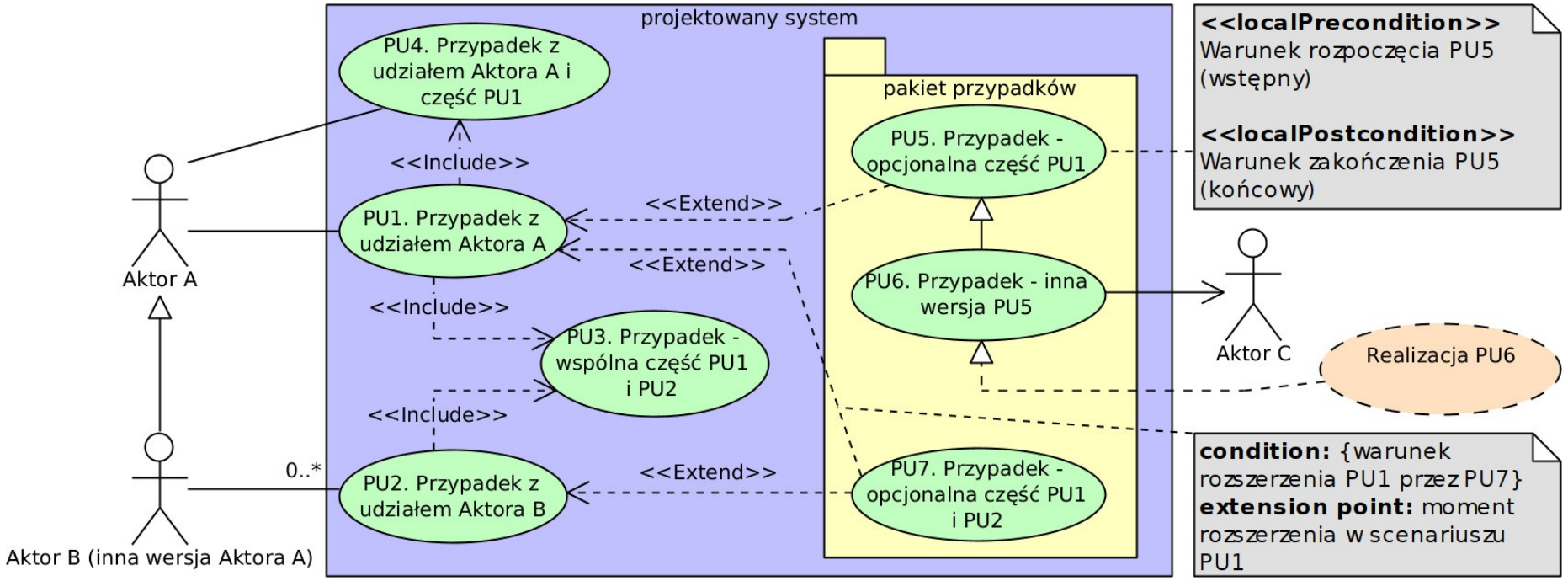
## Przypadek użycia

- Może mieć **warunki wstępne** (*«localPrecondition»*) – spełnienie którego wymagania lub wykonanie którego przypadku użycia pozwala rozpocząć wykonanie tego przypadku użycia.
- Może mieć **warunki końcowe** (*«localPostcondition»*) – spełnienie którego wymagania lub wykonanie którego przypadku użycia pozwala zakończyć wykonanie tego przypadku użycia.



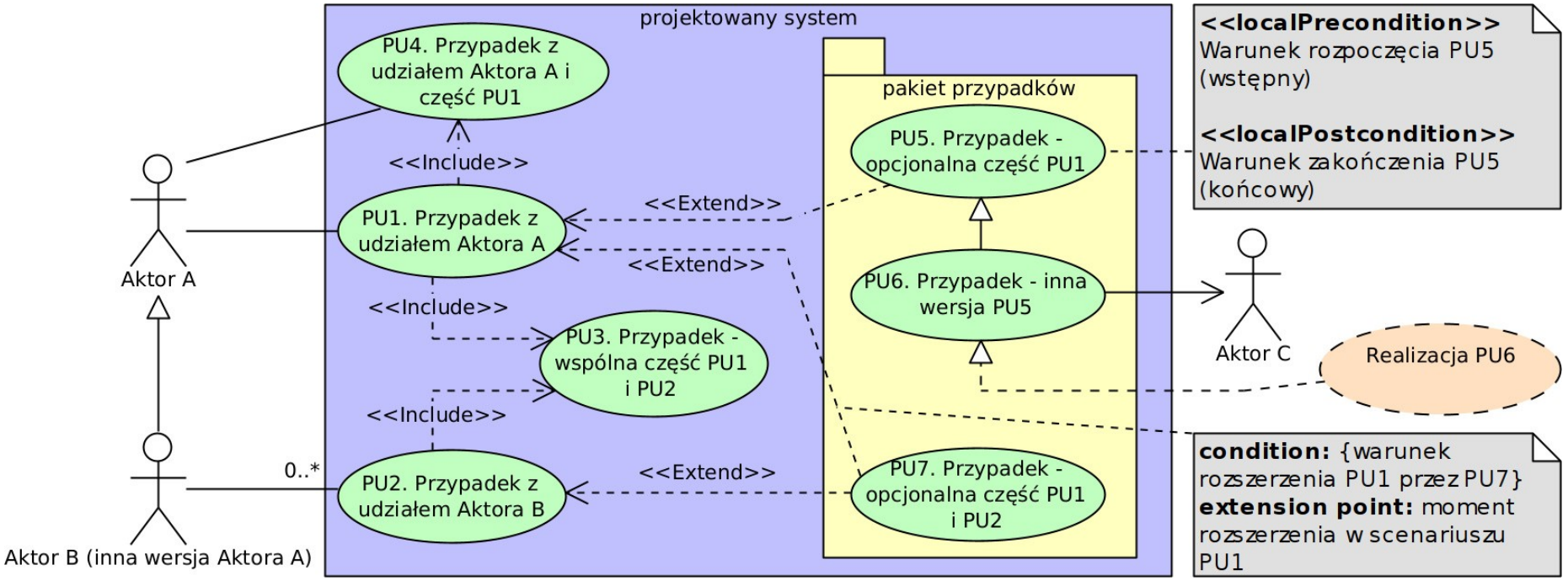
## Aktor /actor/

- Rola użytkownika systemu (człowiek lub inny system).
  - Aktor-system może (nie musi) być modelowany innym „obrazkiem”.
- Ma **bezpośredni lub pośredni udział** w wykonaniu przypadku użycia.
- Ma **pasywny lub aktywny (inicjujący) udział** w wykonaniu przypadku użycia.



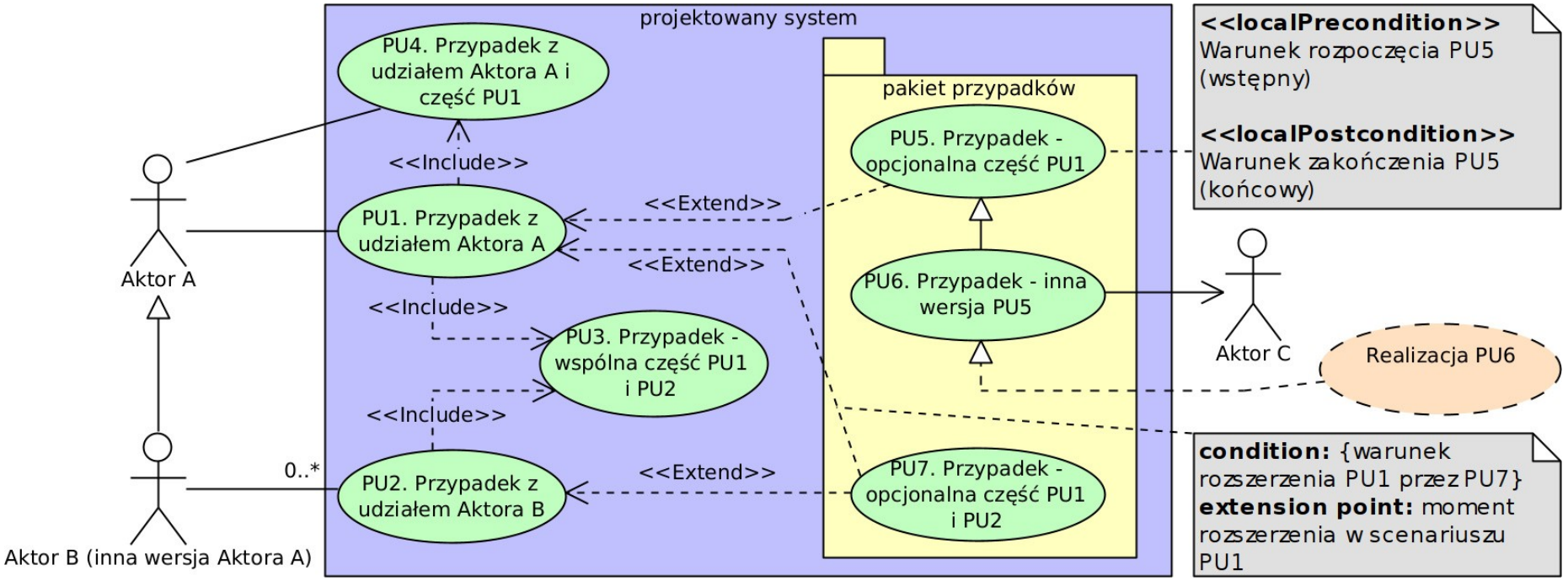
## System /system/

- Modelowany system lub jego podsystem (część innego systemu).
  - Podsystem ma stereotyp *«subsystem»* nad swoją nazwą.
- NIE zawiera innego systemu i NIE zawiera aktorów.
  - Modułową budowę systemu można pokazać na diagramie wdrożenia.
- Obejmuje przypadki użycia modelowanego systemu i ich pakiety.



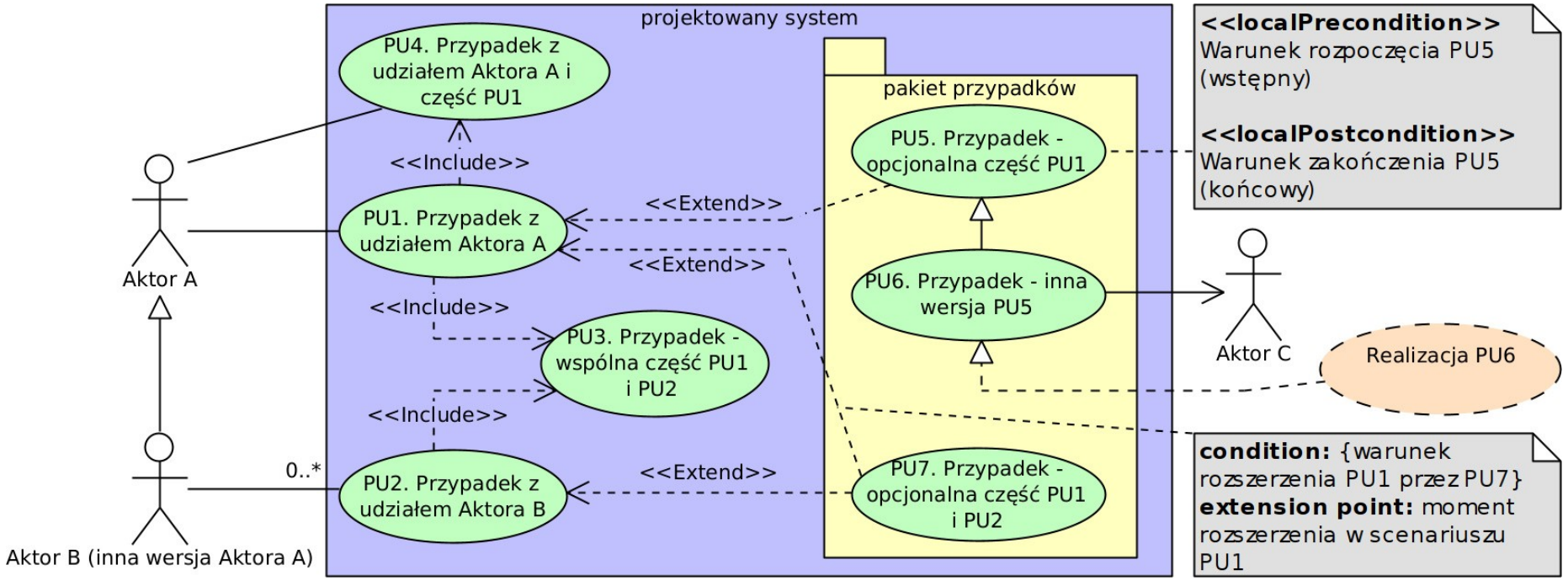
## Relacje powstałe w analizie przypadków użycia

- **Relacja uogólnienia** /generalization/ (ciągła linia z grotem Δ).
  - Łączy przypadki użycia lub aktorów w związek wersja podstawowa (ogólna)-wersja pochodna (szczególna).
  - Grot Δ wskazuje na „wersję podstawową”.



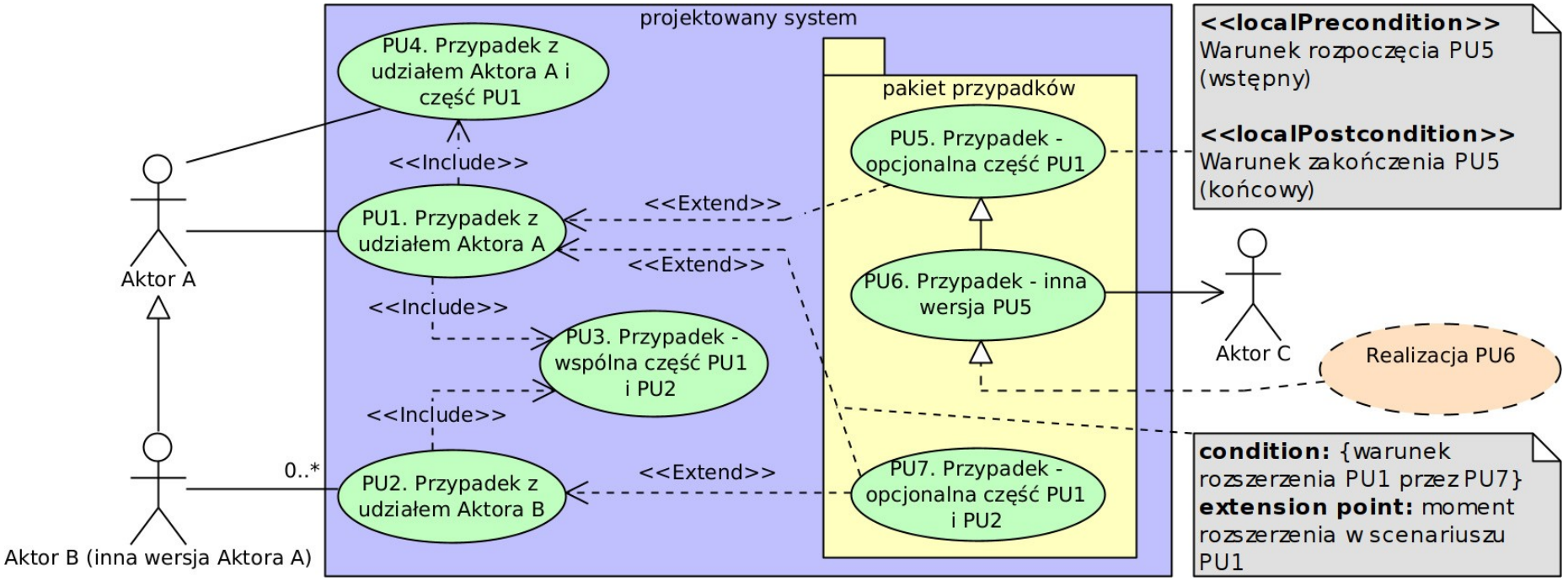
## Relacje powstałe w analizie przypadków użycia

- **Relacja asocjacji** /association/ (ciągła linia z otwartym grotem lub bez).
  - Łączy aktora z przypadkiem użycia, gdy aktor inicjuje wykonanie przypadku lub przypadek bezpośrednio komunikuje się z aktorem.
  - Łączy aktorów ze sobą, gdy komunikują się ze sobą poza systemem.
  - Pokazuje kierunek komunikacji, gdy jest skierowana.
  - Może wyrażać stosunek ilościowy między połączonymi elementami.



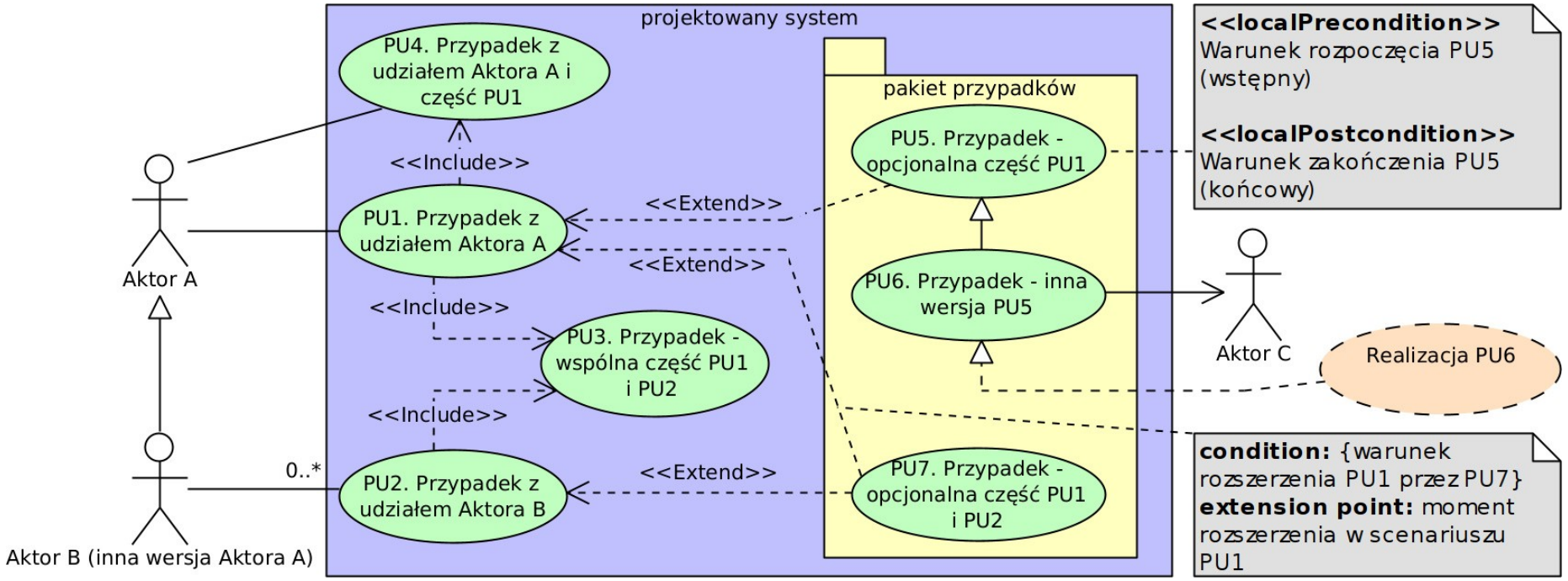
## Relacje powstałe w analizie przypadków użycia

- **Relacja zawierania *«include».***
  - Łączy przypadki użycia w związek zależna całość–obowiązkowa część.
  - Wskazuje na część.
  - Wskazany przypadek użycia obowiązkowo zachodzi w ramach wskazującego przypadku.



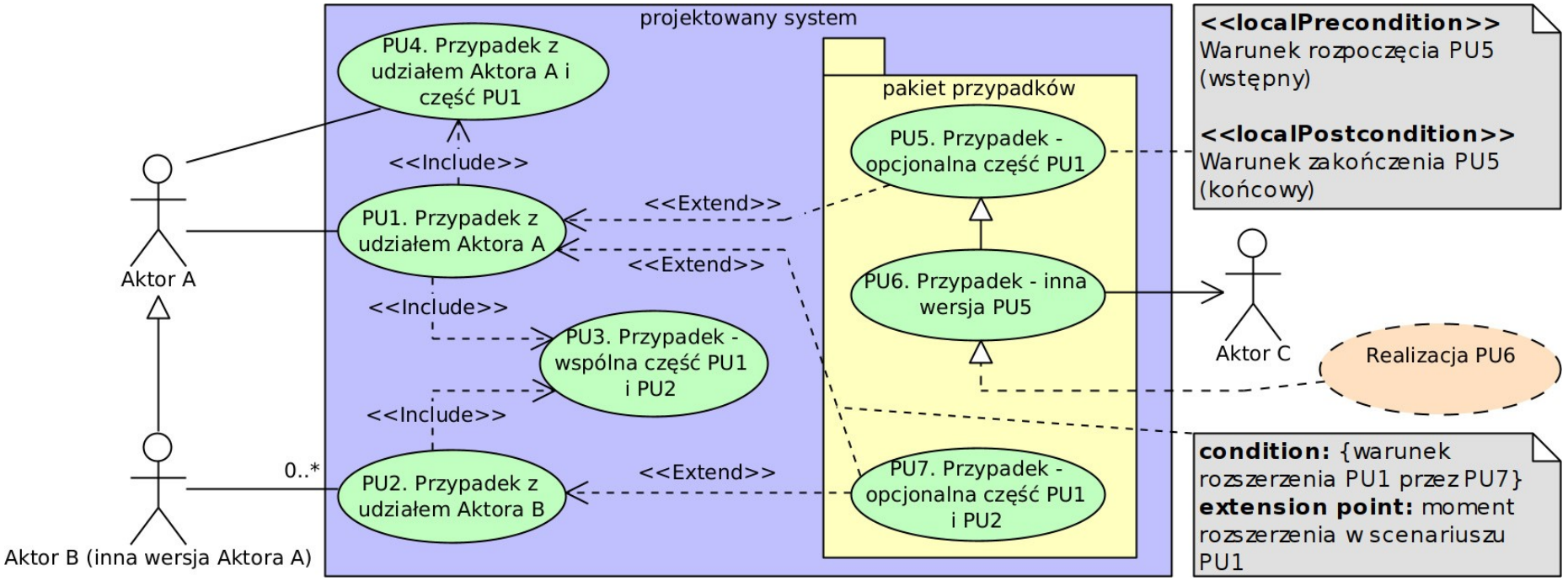
## Relacje powstałe w analizie przypadków użycia

- **Relacja rozszerzania *«extend».***
  - Łączy przypadki użycia w związek niezależna całość–opcjonalna część.
  - Wskazuje na całość.
  - Wskazujący przypadek użycia opcjonalnie lub alternatywnie z innym zachodzi w ramach wskazanego przypadku.
  - Warunek i moment rozszerzenia można podać w notatce i w scenariuszu realizacji rozszerzanego przypadku.



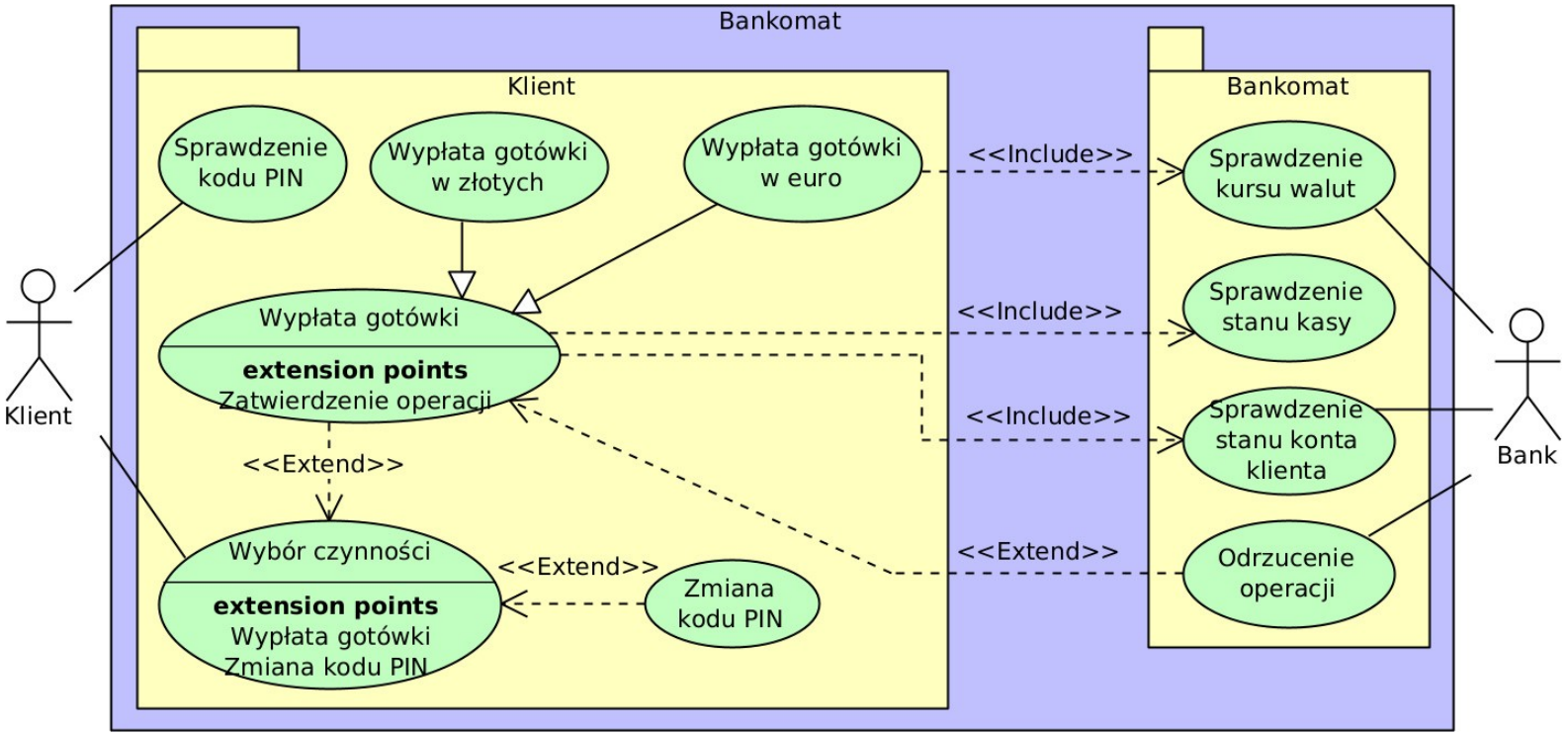
## Relacje powstałe w analizie przypadków użycia

- **Relacja realizacji** /realization/ (przerywana linia z grotem Δ).
  - Łączy przypadek użycia ze współdziałaniem w celu jego realizacji.
  - Współdziałanie można pokazać np. na diagramie klas, gdzie będzie zawierać klasy uczestniczące w realizacji zadań tego przypadku użycia.
  - Grot Δ wskazuje na przypadek użycia.



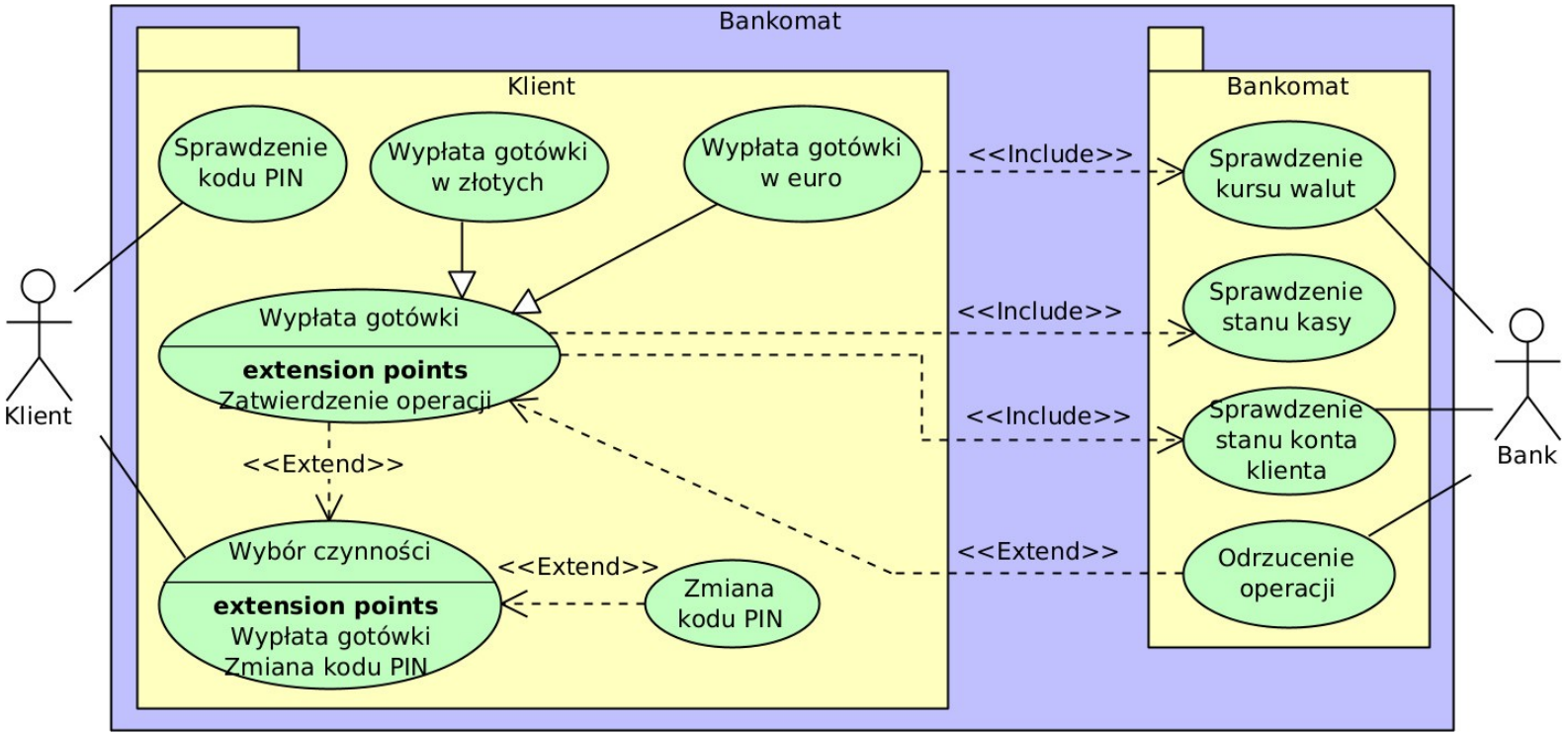
## Przykład dekompozycji przypadków użycia

- Modelowany jest system **Bankomat**.
- Pakiety **Klient** i **Bankomat** grupują przypadki użycia na podstawie udziału w nich aktorów.
- **Klient** jest powiązany tylko z przypadkami użycia, które inicjuje.
- **Bank** jest powiązany tylko z przypadkami użycia, które inicjują z nim komunikację (wymianę informacji).



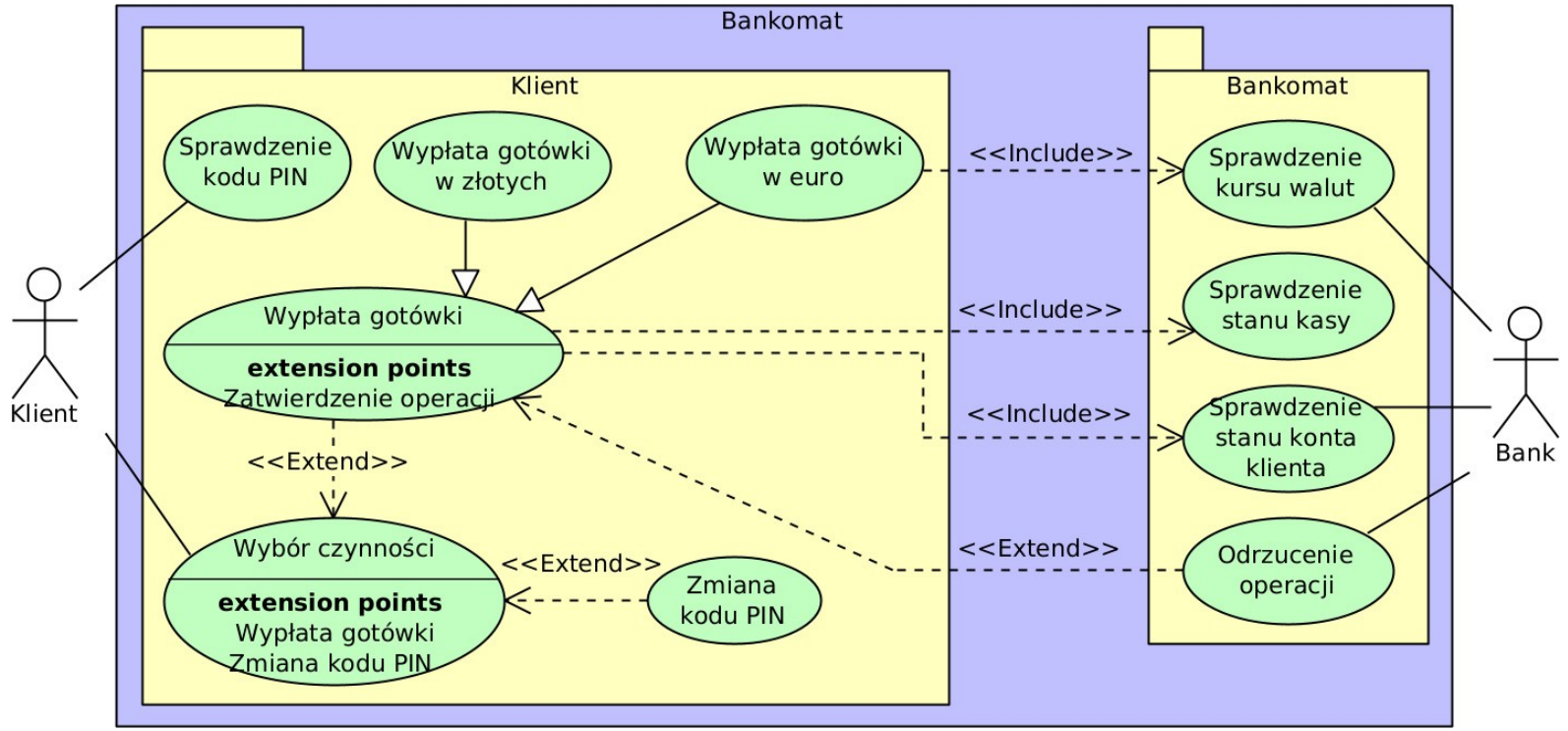
## Przykład dekompozycji przypadków użycia

- Klient inicjuje PU **Sprawdzenie kodu PIN**.
- Następnie (diagram nie może pokazać następstwa) **Klient** inicjuje PU **Wybór czynności**, w którym może (*«extend»*) przejść do wykonania PU **Zmiana kodu PIN** lub PU **Wypłata gotówki**.
- Wykonanie PU **Wypłata gotówki** to (uogólnienie) wykonanie PU **Wypłata gotówki w złotych** lub PU **Wypłata gotówki w euro**.



## Przykład dekompozycji przypadków użycia

- Wykonanie PU **Wypłata gotówki w euro** zawiera (*«include»*) wykonanie PU **Sprawdzenie kursu walut** z udziałem **Banku**.
- Wykonanie PU **Wypłata gotówki** zawiera (*«include»*) wykonanie PU **Sprawdzenie stanu kasy** i PU **Sprawdzenie stanu konta klienta** z udziałem **Banku**.
- Wykonanie PU **Wypłata gotówki** może zawierać (*«extend»*) wykonanie PU **Odrzucenie operacji** z udziałem **Banku**.



## Dokumentacja przypadku użycia

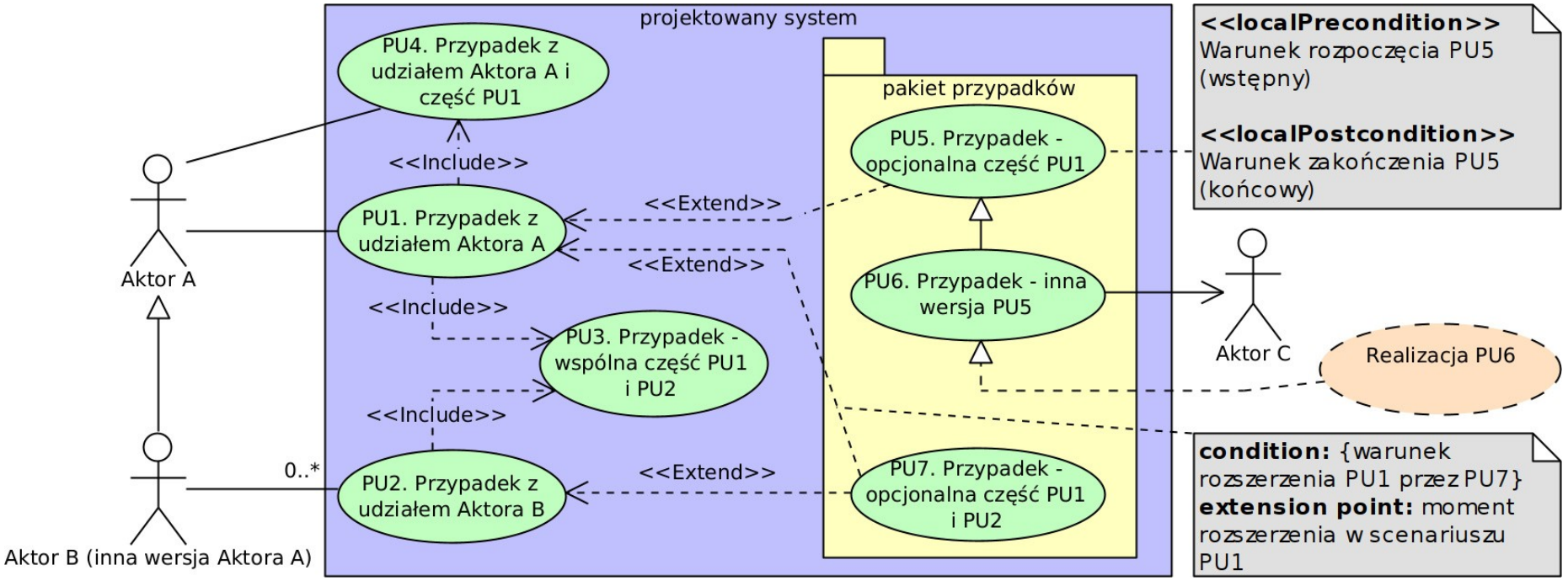
- Typowy skład słownego opisu przypadku użycia (PU):
  - **numer PU**;
  - **nazwa PU** – krótka, w formie wykonywanej operacji, a NIE obiektu;
  - **cel PU** – dłuższe wyjaśnienie (jeśli nazwa nie jest dość precyzyjna);
  - **warunki wstępne PU** – wymagania i przypadki użycia pozwalające rozpocząć wykonanie PU;
  - **warunki końcowe PU** – wymagania i przypadki użycia pozwalające zakończyć wykonanie PU (po czym poznać, że wykonał się prawidłowo);
  - **przypadki testowania** – testy sprawdzające wykonanie PU;
  - **scenariusz PU** – przepływ zdarzeń, które składają się na wykonanie PU (algorytm wykonania PU).
- Scenariusz jest najważniejszy i obowiązkowy!

## Scenariusz realizacji przypadku użycia

- Przepływ zdarzeń, które składają się na wykonanie przypadku użycia w modelowanym systemie.
- Uwzględnia:
  - aktorów,
  - artefakty (przepływ obiektów),
  - zdarzenia – czynności wykonywane w przez aktorów w modelowanym systemie lub przez modelowany system (przepływ sterowania).
- Może być tekstowy (pseudokod) lub graficzny (diagram czynności).
- Oprócz głównego scenariusza mogą być alternatywne scenariusze.
- Pokazuje, w którym momencie następuje rozszerzenie PU przez inny przypadek użycia, będący z nim w relacji *«include».*
- Pokazuje, w którym momencie i pod jakim warunkiem lub alternatywnie z jakim innym przypadkiem użycia następuje rozszerzenie PU przez inny przypadek użycia, będący z nim w relacji *«extend».*

## Przykład dokumentacji przypadku użycia

- **Opis sytuacji:**
  - PU1 zawiera w sobie PU3 i PU4 (relacje «include»).
  - PU1 jest rozszerzany przez PU5 (lub PU6) lub alternatywnie przez PU7 (relacje «extend»).
  - PU2 zawiera w sobie PU3 (relacja «include»).
  - PU2 może być rozszerzany przez PU7 (relacja «extend»).



## Przykład dokumentacji przypadku użycia

- **Dokumentacja PU1:**
  - **numer:** PU1
  - **nazwa:** Przypadek z udziałem aktora A
  - **cel:** Aktor A bierze udział w wykonaniu PU1, aby otrzymać pewną informację.
  - **warunki wstępne:** W roli Aktora A nie występuje Aktor B.
  - **warunki końcowe:** Wykonano PU5/PU6 i Aktor A dostał informację lub PU7.
  - **scenariusz**:

    (1) Aktor A inicjuje wykonanie PU1.

    (2) System przygotowuje wstępną treść informacji.

    (3) Wykonanie PU3 w celu uzupełnienia tej informacji.

    (4) Wykonanie PU4 w celu sprawdzenia tej informacji.

    (5) System przedstawia Aktorowi A tę informację do zatwierdzenia.

    (6) Wykonanie PU5, jeśli Aktor A chce zatwierdzić tę informację, lub wykonanie PU7, jeśli Aktor A chce ją odrzucić. Zamiast PU5 można wykonać PU6, jeśli Aktor C ma wiedzieć o zatwierdzeniu tej informacji.

## Przykład dokumentacji przypadku użycia

- **Dokumentacja PU2:**
  - **numer:** PU2
  - **nazwa:** Przypadek z udziałem aktora B
  - **cel:** Aktor B bierze udział w wykonaniu PU2, aby odrzucić błędną informację.
  - **warunki wstępne:** Aktor A wykonał PU1 razem z PU5 lub wykonał PU6 i utworzył informację.
  - **warunki końcowe:** Dla błędnej informacji wykonano PU7 i odrzucono ją.
  - **scenariusz PU**:
    (1) Aktor B inicjuje wykonanie PU2.
    (2) System przedstawia treść informacji.
    (3) Wykonanie PU4 w celu sprawdzenia tej informacji.
    (4) System przedstawia Aktorowi B tę informację do zatwierdzenia.
    (5) Wykonanie PU7, jeśli ta informacja jest błędna i Aktor B chce ją odrzucić.

- Co robi system i co robią aktorzy w ramach wykonania PU3–PU7?
  To pokazują ICH dokumentacje.

# 3

# Modelowanie wymagań i przypadków użycia

## Wymagania funkcjonalne a przypadki użycia

- **Wymagania funkcjonalne:**
  - są szczegółowe,
  - skupiają się na funkcjonalności, celach i zachowaniu systemu,
  - pomagają zdefiniować zadania projektu programistycznego,
  - są podstawą modelowania i testowania systemu,
- **Przypadki użycia:**
  - są ogólne,
  - skupiają się na zadaniach i potrzebach użytkowników,
  - pomagają zdefiniować przepływy sterowania i danych w projekcie programistycznym,
  - są podstawą weryfikacji (zgodność ze specyfikacją) i walidacji (zgodność z oczekiwaniami klienta) działania systemu,
  - pomagają zaprojektować interfejs użytkownika systemu.

## Wymagania funkcjonalne a przypadki użycia

- **Wymagania funkcjonalne pomagają zdefiniować przypadki użycia:**
  - wymaganie funkcjonalne opisuje cele systemu,
  - przypadek użycia opisuje drogę do celu,
  - klient i analityk myślą celowo.
- **Przypadki użycia pomagają zdefiniować wymagania funkcjonalne:**
  - przypadek użycia opisuje proces biznesowy,
  - wymaganie funkcjonalne opisuje cele systemu,
  - klient i analityk myślą zadaniowo.

## Elementy modelowania wymagań

na podst. UML Przewodnik użytkownika, G. Booch, J. Rumbaugh, I. Jacobson; wyd. WNT

1. **Wyznacz granice systemu w jego środowisku:**
   - Zdefiniuj aktorów związanych z systemem:
     - są to użytkownicy korzystający z jego usług, zarządzający nim, pielęgnujący go itp.;
     - nazwa aktora – rodzaj użytkownika systemu;
     - stereotyp aktora (np. «database») – rodzaj aktora.
   - Wykonaj analizę wspólności i zmienności aktorów:
     - utwórz ogólniejszego aktora dla aktorów, których część powiązań z systemem jest identyczna;
     - powiąż aktorów relacją uogólnienia, jeśli jeden z nich jest ogólniejszą wersją drugiego.
   - Powiąż aktorów z dotyczącymi ich przypadkami użycia:
     - ale nie wiąż aktora z przypadkiem użycia, jeśli ogólniejsza wersja tego aktora już jest powiązana z tym przypadkiem użycia.

## Elementy modelowania wymagań

na podst. UML Przewodnik użytkownika, G. Booch, J. Rumbaugh, I. Jacobson; wyd. WNT

**2. Zdefiniuj wymagania systemu:**

- Zdefiniuj otoczenie systemu, czyli jego aktorów.
- Zdefiniuj przypadki użycia:
  - są to działania, które aktorzy oczekują od systemu;
  - nazwa przypadku użycia – opis działania.
- Dodaj do przypadku użycia notatkę o dotyczących go wymaganiach niefunkcjonalnych.
- Wykonaj analizę wspólności i zmienności przypadków użycia:
  - utwórz ogólniejszy przypadek użycia dla przypadków użycia, których działanie jest prawie identyczne;
  - powiąż przypadki użycia relacją uogólnienia, jeśli jeden z nich jest ogólniejszą wersją drugiego.

## Elementy modelowania wymagań

na podst. UML Przewodnik użytkownika, G. Booch, J. Rumbaugh, I. Jacobson; wyd. WNT

**2. Zdefiniuj wymagania systemu (c.d.):**

- Wykonaj analizę wspólności i zmienności przypadków użycia (c.d.):
  - Utwórz nowy przypadek użycia dla działania:
    - które jest wspólną częścią działania różnych przypadków użycia,
    - lub które jest częścią działania któregoś przypadku użycia, ale może też być samodzielna;
    - nowy, wydzielony przypadek użycia połącz z jego źródłowym przypadkiem użycia relacją zawierania («include»).
  - Utwórz nowy przypadek użycia dla działania:
    - które jest opcjonalną częścią działania któregoś przypadku użycia,
    - lub które jest alternatywną (względem innej) częścią działania któregoś przypadku użycia;
    - nowy, wydzielony przypadek użycia połącz z jego źródłowym przypadkiem użycia relacją rozszerzania («extend»).

## Elementy modelowania wymagań

na podst. UML Przewodnik użytkownika, G. Booch, J. Rumbaugh, I. Jacobson; wyd. WNT

**3. Opisz każdy przypadek użycia:**

- Opisz cel jego realizacji – uzasadnienie.
- Opisz warunki początkowe i końcowe jego realizacji.
- Opisz główny i alternatywne scenariusze jego realizacji – sekwencje zdarzeń w jego działaniu:
  - przepływ czynności aktorów i systemu,
  - przepływ danych między aktorami a systemem,
  - tekstowo (w języku naturalnym) i / lub graficznie (diagramem czynności).

## Elementy modelowania wymagań

na podst. UML Przewodnik użytkownika, G. Booch, J. Rumbaugh, I. Jacobson; wyd. WNT

**3. Opisz każdy przypadek użycia (c.d.):**

- Zdefiniuj przypadki jego testowania – testy systemu:
  - wdrożeniowy – czy przypadek użycia został prawidłowo wdrożony;
  - akceptacyjny – czy jego implementacja zadowala klienta;
  - funkcjonalny – czy jego implementacja jest zgodna z jego opisem;
  - w odniesieniu do jednego, związanego z nim aktora;
  - określ stan początkowy systemu i dane wejściowe testu;
  - określ pożądany stan końcowy systemu i dane wyjściowe testu.

# Proces identyfikacji wymagań

na podst. Inżynieria Oprogramowania, Z. Kruczkiewicz, PWr

| Produkty wejściowe - czynności | Produkty wyjściowe | Opis produktu wyjściowego |
|---|---|---|
| **lista kandydujących wymagań** | **Lista znamionowa** | status, szacowany koszt, priorytety, poziom ryzyka implementacji itp. |
| **zrozumienie kontekstu systemu** (**wykład 3**) | ***Model dziedziny)**** (*domain model*)-najważniejsze obiekty systemu: „rzeczy” lub zdarzenia podawane przez ekspertów | **diagram najważniejszych klas dziedziny** (*domain classes*) z niewielką ilością operacji-metod (około 10-50 w notacji UML), reszta przewidywanych klas w glosariuszu (*glossary*); |
| | ***Model biznesowy**** (*business model*) - wewnętrzny model procesu biznesowego organizacji, wyszczególniany przez stronę zamawiającą systemu (*customers)* | **„business use case” :**<br>**a) Opis przypadków użycia („uses cases”) i aktorów („actors”)** odpowiadających procesowi biznesowemu oraz klientom procesu biznesowego<br>**b) biznesowy model obiektowy** (*business object model*) składający się z wykonawców (*workers*), encji biznesowych (*business entities*), jednostek pracy (*work units*) z „use case” |

# Proces specyfikacji wymagań

na podst. Inżynieria Oprogramowania, Z. Kruczkiewicz, PWr

| | | |
|---|---|---|
| **funkcjonalne wymagania** | ***Model przypadków użycia* (identyfikacja przypadków użycia z modelu biznesowego)** | Proces reprezentowania wymagań jako przypadków użycia UML oraz innych produktów:<br>**1. opis tekstowy** realizujących zachowanie systemu przy działaniu poszczególnych przypadków użycia - czyli opis sekwencji akcji odpowiednich do modyfikacji, przeglądu, projektowania i testowania,<br>**2. model przypadków użycia zawierający aktorów i przypadki użycia oraz powiązania** (np. dziedziczenia) między nimi oraz: **dodatkowo diagram** czynności modelujący scenariusz przypadku użycia – **wykład 3**<br>**3. opis architektury** przypadków użycia<br>**4. glosariusz** - definicje ważnych pojęć wyprowadzanych z modelu dziedziny lub modelu biznesowego,<br>**5. prototyp interfejsu użytkownika** - interakcje między aktorami - ludźmi i oprogramowaniem |
| **niefunkcjonalne wymagania** | **uzupełniające wymagania lub indywidualne wymagania** | **1. specjalne wymagania** zawierające niefunkcjonalne wymagania w postaci opisu tekstowego<br>**2. ograniczenia środowiska i implementacji** (np. typ komputera, typ plików, rodzaj systemu operacyjnego, typ oprogramowania Internetu), zależności, konserwacja, zdolność do poszerzania, |

## Uwaga!

- **Definiowanie wymagania i przypadku użycia musi być zrozumiałe, ustandaryzowane i wewnętrznie spójne.**
  - ułatwia to współpracę z klientem
  - oraz komunikację wewnątrz zespołu projektowego (analitycy, projektanci, programiści, testerzy…).
- **Definicja wymagania i przypadku użycia powinna być możliwie jak najprostsza:**
  - zwłaszcza w ocenie użytkowników systemu.
- **Definicja wymagania i przypadku użycia z perspektywy użytkownika** jest lepsza niż z perspektywy systemu.
- **Przypadki użycia NIE definiują:**
  - wyglądu interfejsu użytkownika systemu,
  - danych przetwarzanych przez system.
- **Diagram przypadków użycia NIE pokazuje kolejności realizacji przypadków użycia.**

# 4

# Przykłady

## Przykłady

zobacz: Inżynieria Oprogramowania, Z. Kruczkiewicz, PWr, wykład 2

1. Identyfikacja przypadków użycia na diagramie przypadków użycia na podstawie zdefiniowanych wymagań… (strony 32—38).
2. Identyfikacja i specyfikacja przypadków użycia na diagramie przypadków użycia. System sporządzania rachunków. (strony 39—53).
3. Identyfikacja i specyfikacja przypadków użycia na diagramie przypadków użycia. Biblioteka. (strony 55—71).
4. Identyfikacja i specyfikacja przypadków użycia na diagramie przypadków użycia. Wypożyczalnia książek. (strony 72—79).

Temat następnej prezentacji

# Modelowanie zachowania – diagram czynności