MODELOWANIE STRUKTURY

(Wykład na podstawie literatury: M.Śmiałek „Zrozumieć UML 2.0”, Helion 2005)

Prezentacja struktury na dwóch poziomach: klas i obiektów

(Na diagramach opisujących strukturę fragmentu modelowanej rzeczywistości lub systemu pojawią się zarówno klasy jak i obiekty)

Aspekty modelowanej dziedziny: (wykorzystując klasy i związki m.n.)

→ **Zbiór wszystkich możliwych stanów obiektów klas**

→ **Zbiór wszystkich usług dostarczanych przez klasy**

→ **Zbiór wszystkich możliwych powiązań obiektów klas pomiędzy sobą**

→ **Zbiór wszystkich możliwych ścieżek komunikacji między obiektami klas**

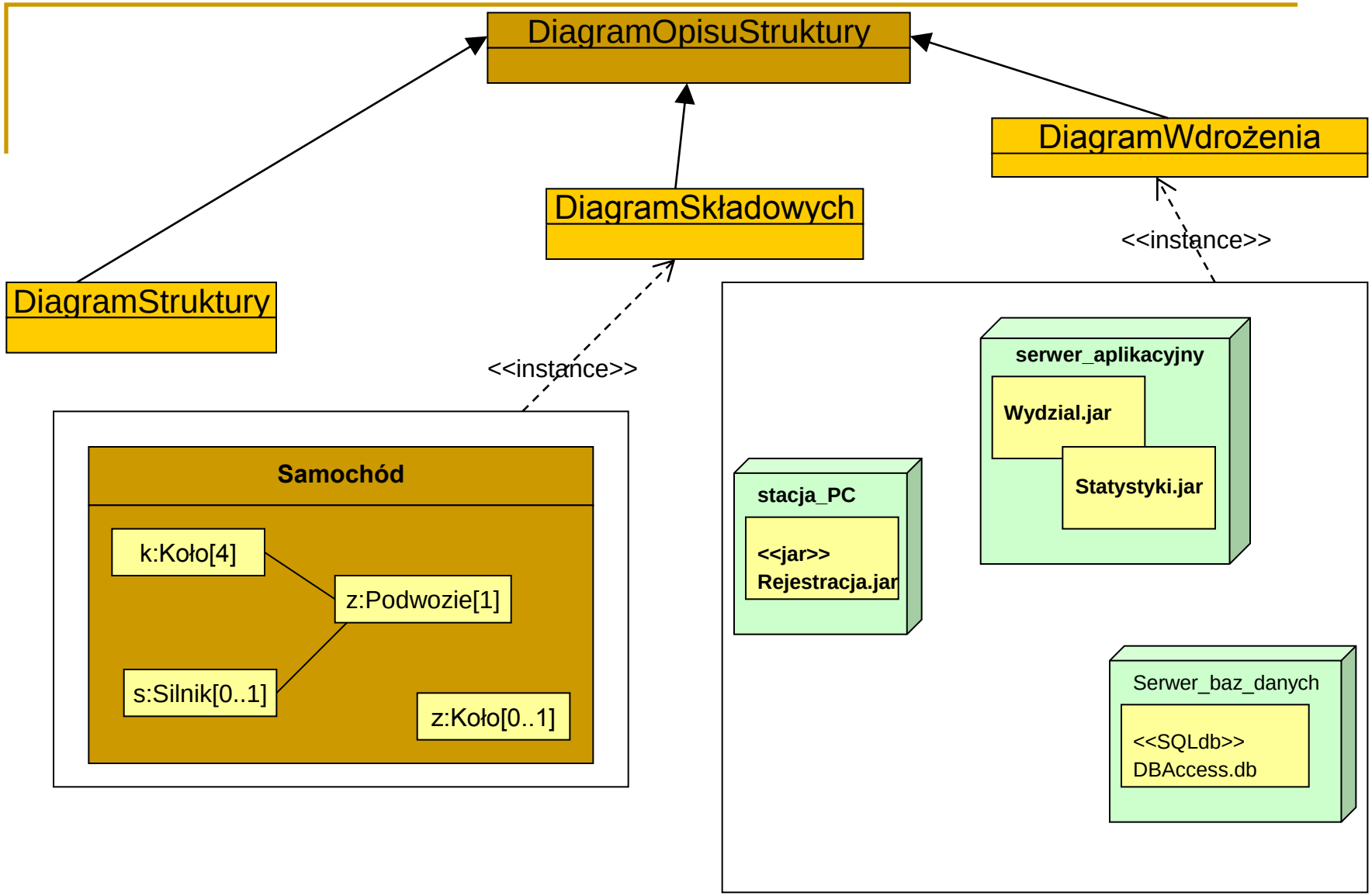
(Diagramy klas pokazują potencjalne możliwości wchodzenia obiektów w interakcje ze sobą. Model klas stanowi słownik dziedziny.

Tworzenie struktury z wykorzystaniem abstrakcji

(Zamykanie wielu klas i obiektów w „pudełku” i traktowanie pudełka jako nowego pojęcia → pakietu (gromadzą wiele klas) lub mogą stanowić całe podsystemy dostarczające na zewnątrz określonej funkcjonalności → komponenty

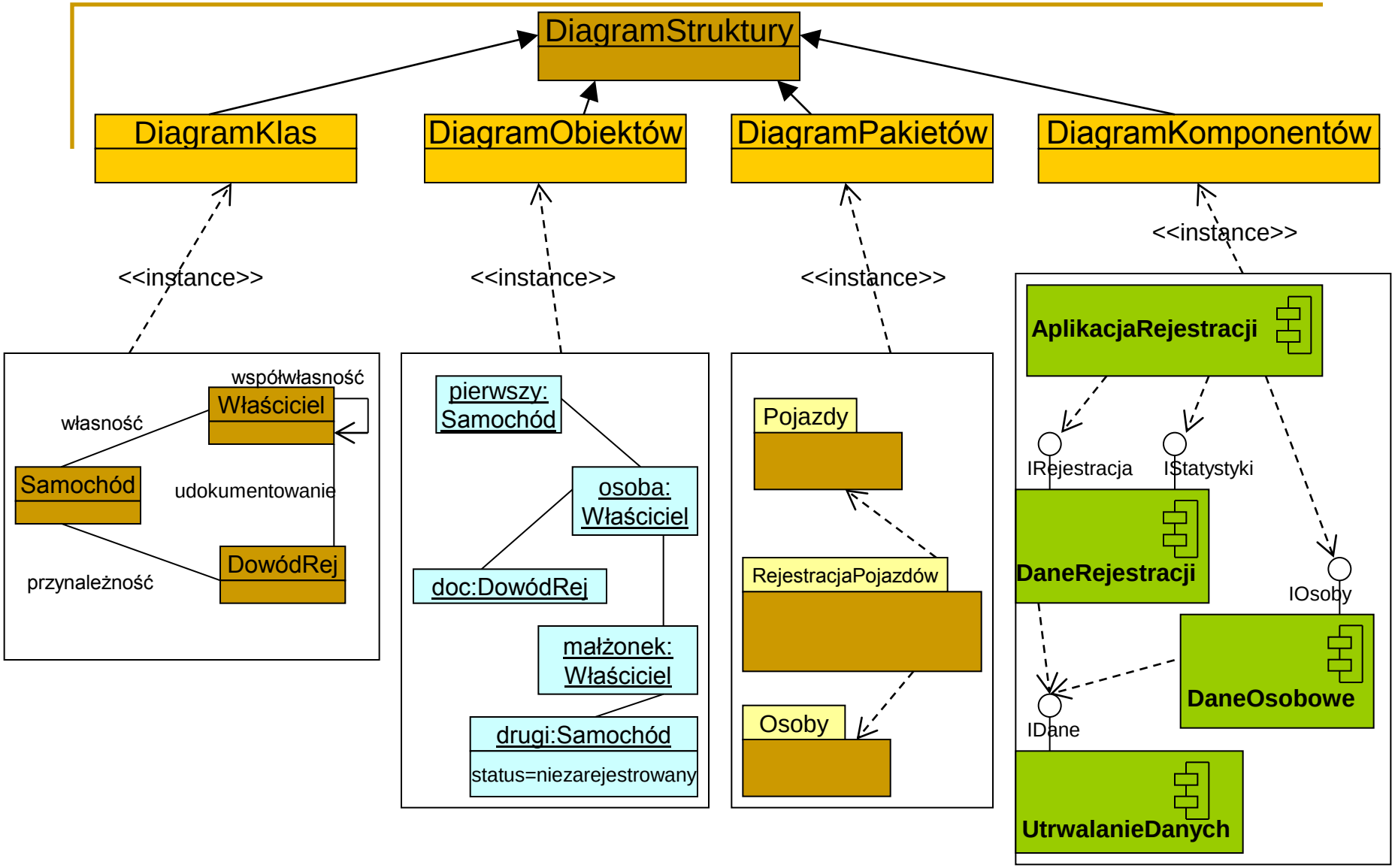
MODELOWANIE STRUKTURY

Diagramy STRUKTURY



MODELOWANIE STRUKTURY

Diagramy STRUKTURY



MODELOWANIE DYNAMIKI

Prezentacja Systemu w działaniu

(Realizacja dynamiki systemu obejmuje przede wszystkim spełnienie rzeczywistych potrzeb zamawiajacego)

Aspekty modelowanej funkcjonalności systemu:

→ **Prezentacja następstwa czasów**

→ **Prezentacja dialogu użytkownika z systemem**

→ **Powiązanie modelu dynamiki z modelem struktury** ***(obiekty z diagramów opisujących dynamikę mają swoje odpowiedniki w obiektach (klasach) prezentowanych na diagramach opisu struktury)***

→ **Diagramy przypadków użycia** ***(use case diagram)*** ***– prezentują jednostki funkcjonalności systemu i ich użytkowników***

→ **Diagramy sekwencji** ***(sequence diagram)*** ***– pokazują ciąg interakcji zachodzącymi pomiędzy obiektami systemu*** ***oraz*** **Diagramy komunikacji** ***(różnica pomiędzy diagramami polega na sposobie prezentacji wymiany komunikatów)***

→ **Diagramy czynności** ***(activity diagram)*** ***– graf opisujący czynności podejmowane w zależności od warunków zewnętrznych i stanu systemu***

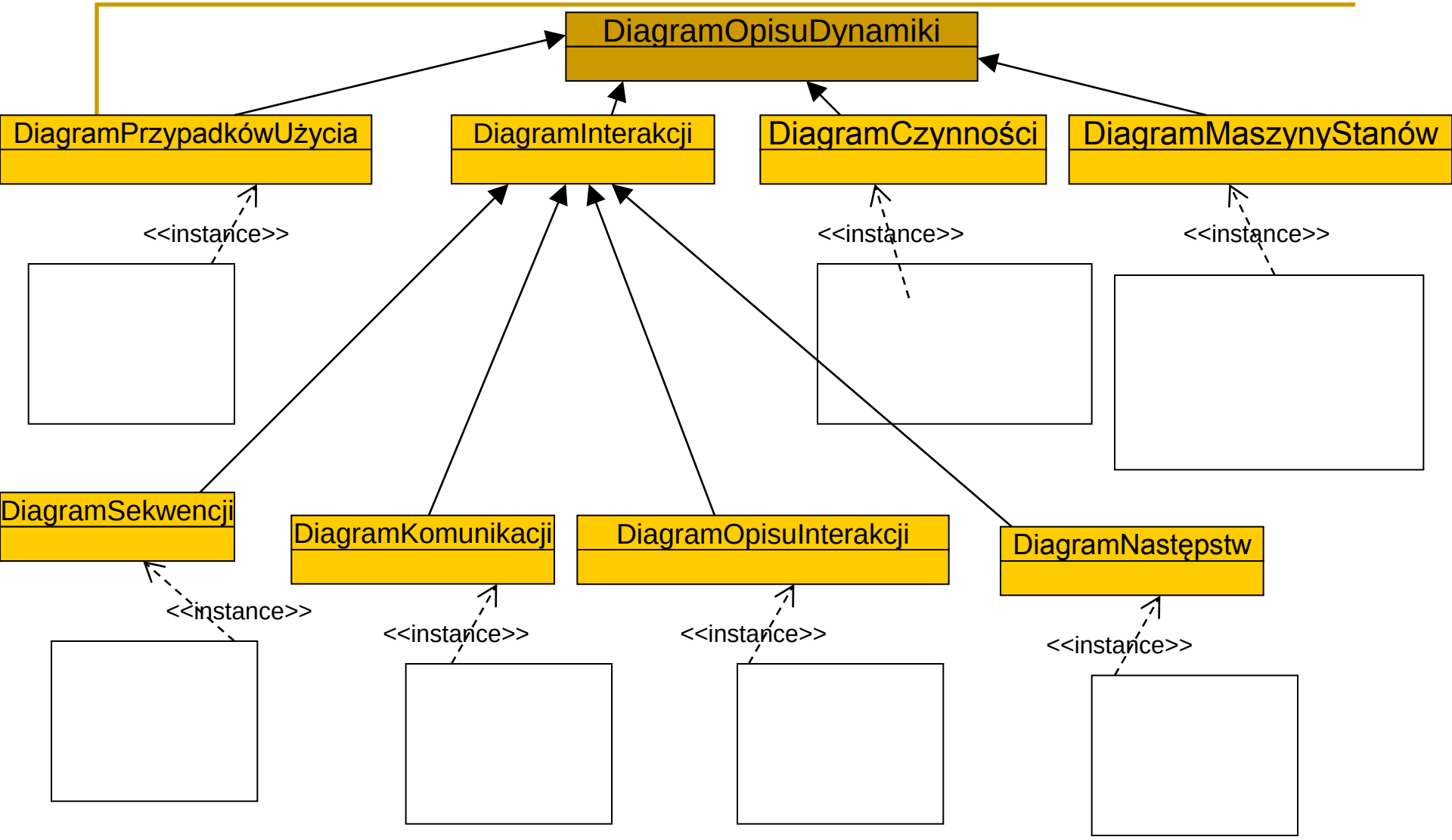
MODELOWANIE DYNAMIKI

→ **Diagramy maszyny stanów *(state machine diagram)*** ***– prezentacja możliwych stanów jakiegoś elementu, klasy z siecią przejść pomiędzy stanami***

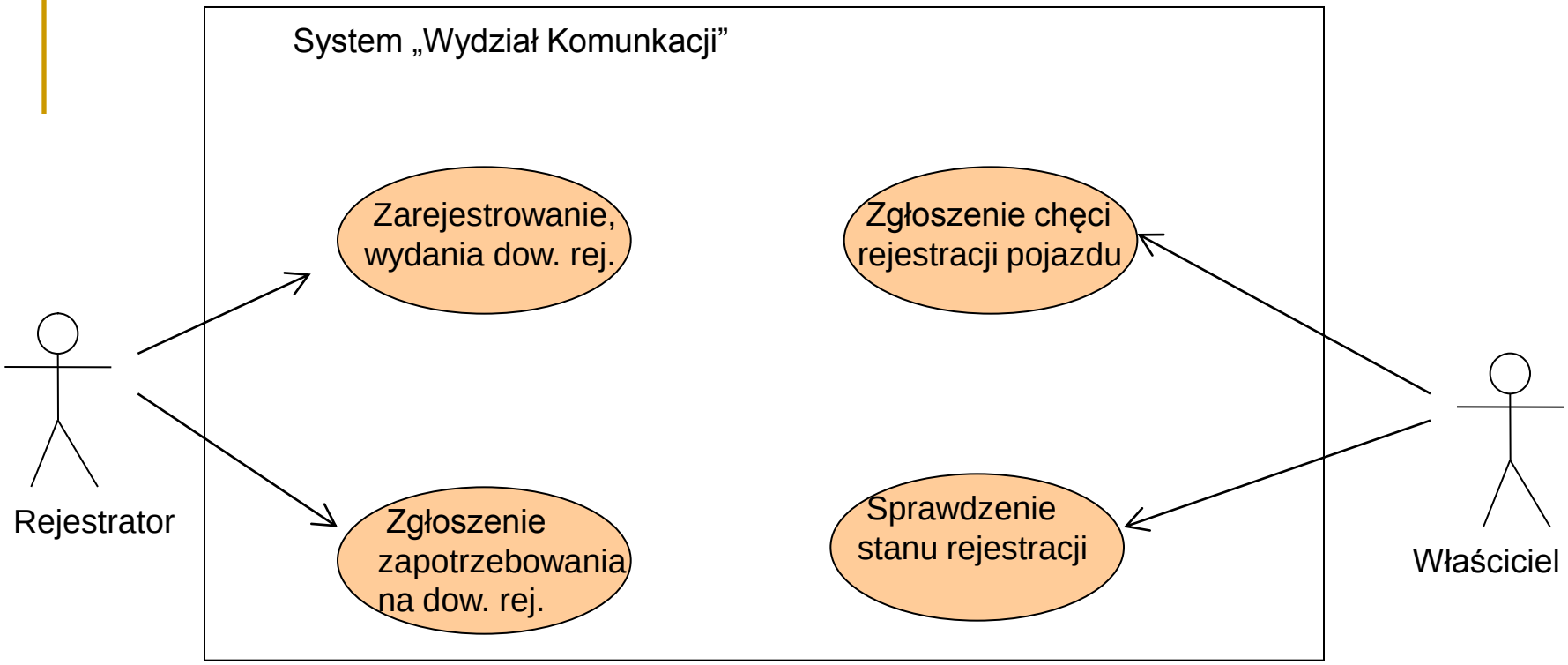
→ **Diagramy opisu interakcji *(interaction overview diagram)*** ***– pokazują interakcje połączone jako sieć czynności***

→ **Diagramy następstw *(timing diagram)*** ***– prezentacja protokołów jak ciągów czasowo uzależnionych komunikatów wymienianych między obiektami.***

MODELOWANIE DYNAMIKI



MODELOWANIE DYNAMIKI
diagram przypadków użycia



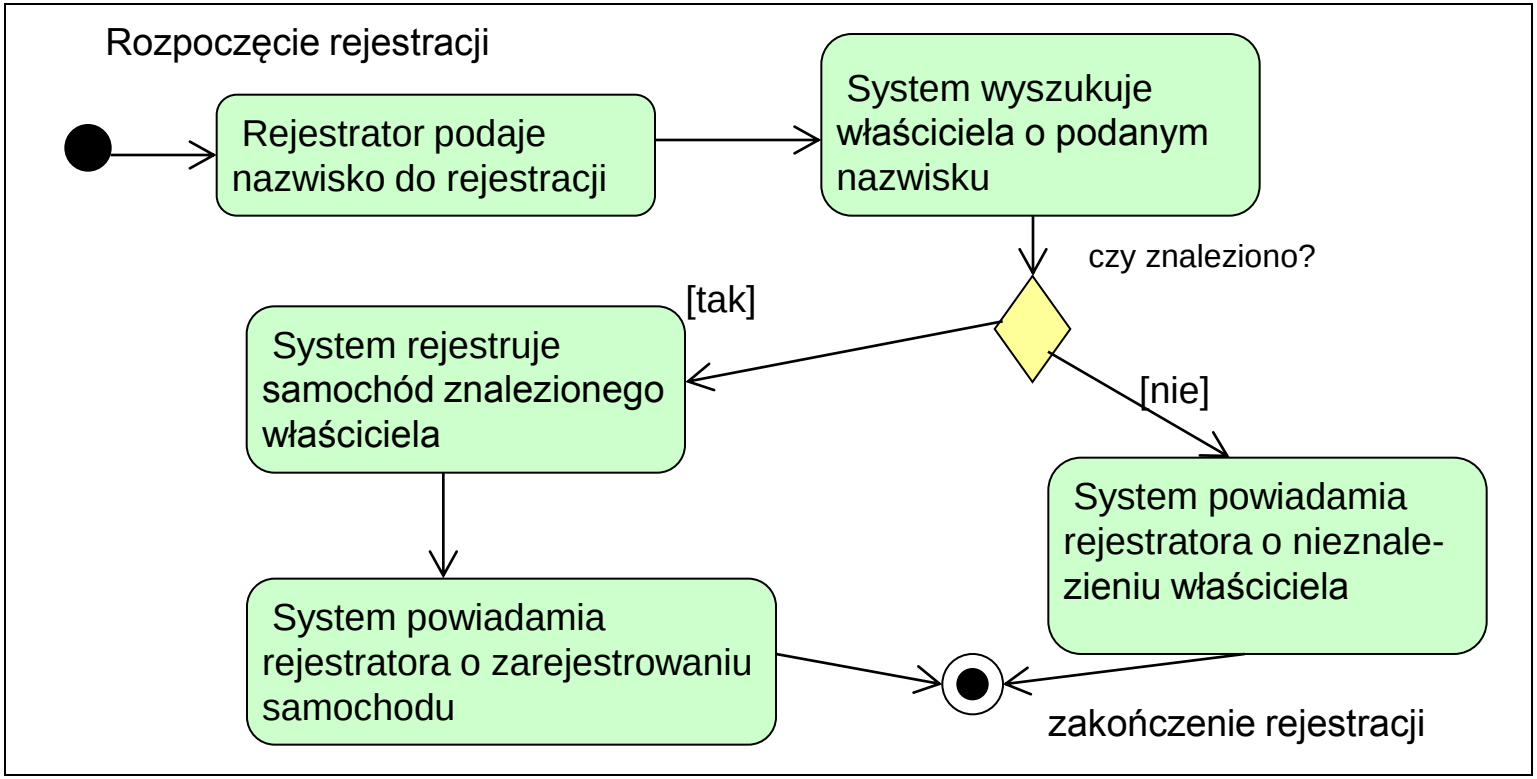
→ **Przypadki użycia – jednostki opisu dynamiki systemu – zestawy scenariuszy**

→ **Scenariusze – sekwencja interakcji podlegających określonym warunkom i prowadzącym do określonego celu (w opisie prezentowana jako tekst lub diagramy czynności)**

→ **Actor – obiekt (klasa obiektów) na zewnątrz systemu, często użytkowników systemu**

MODELOWANIE DYNAMIKI

diagram czynności



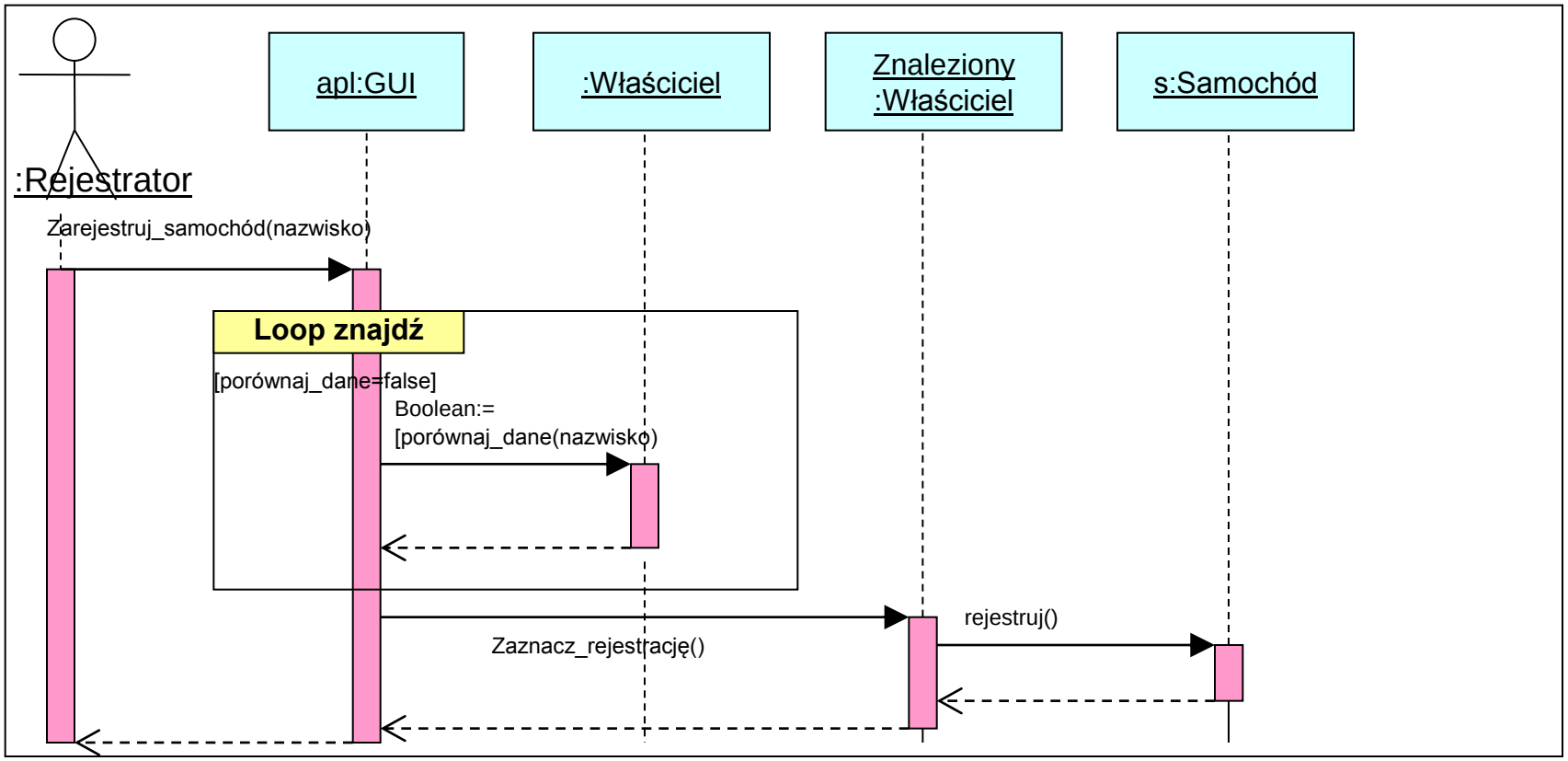
→ **Węzeł początkowy i końcowy – ograniczniki sieci akcji**

→ **Akcje – określone czynności opisane w zaokrąglonych prostokątach**

→ **Węzły decyzyjne – romby z opisem decyzji**

→ **Przepływy sterowania – strzałki, mogą być z uwarunkowaniem**

MODELOWANIE DYNAMIKI
diagram interakcji (sekwencji)



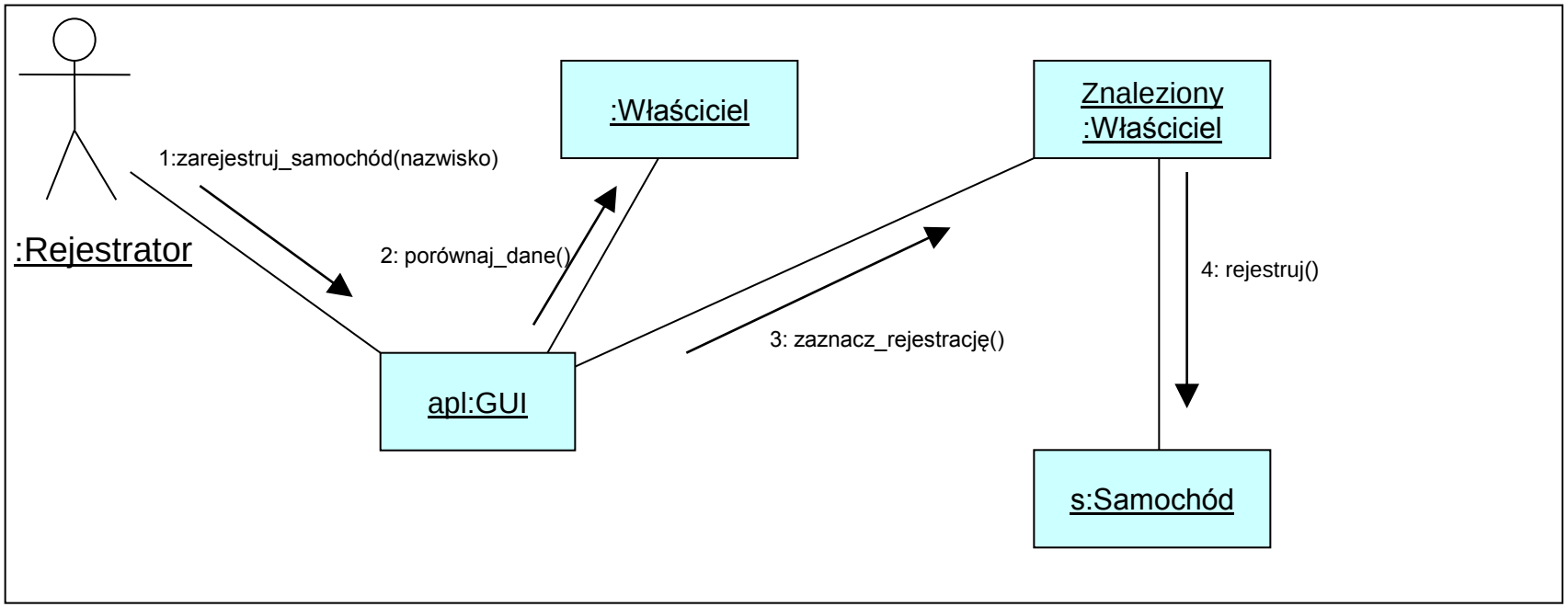
→ Pokazują następstwo czasowe w sekwencji komunikatów

→ Linie życia – pionowe kolumny odnoszace się do obiektów z diagramów obiektów

→ Fragment włączony – sekwencje komunikatów realizowanych iteracyjnie

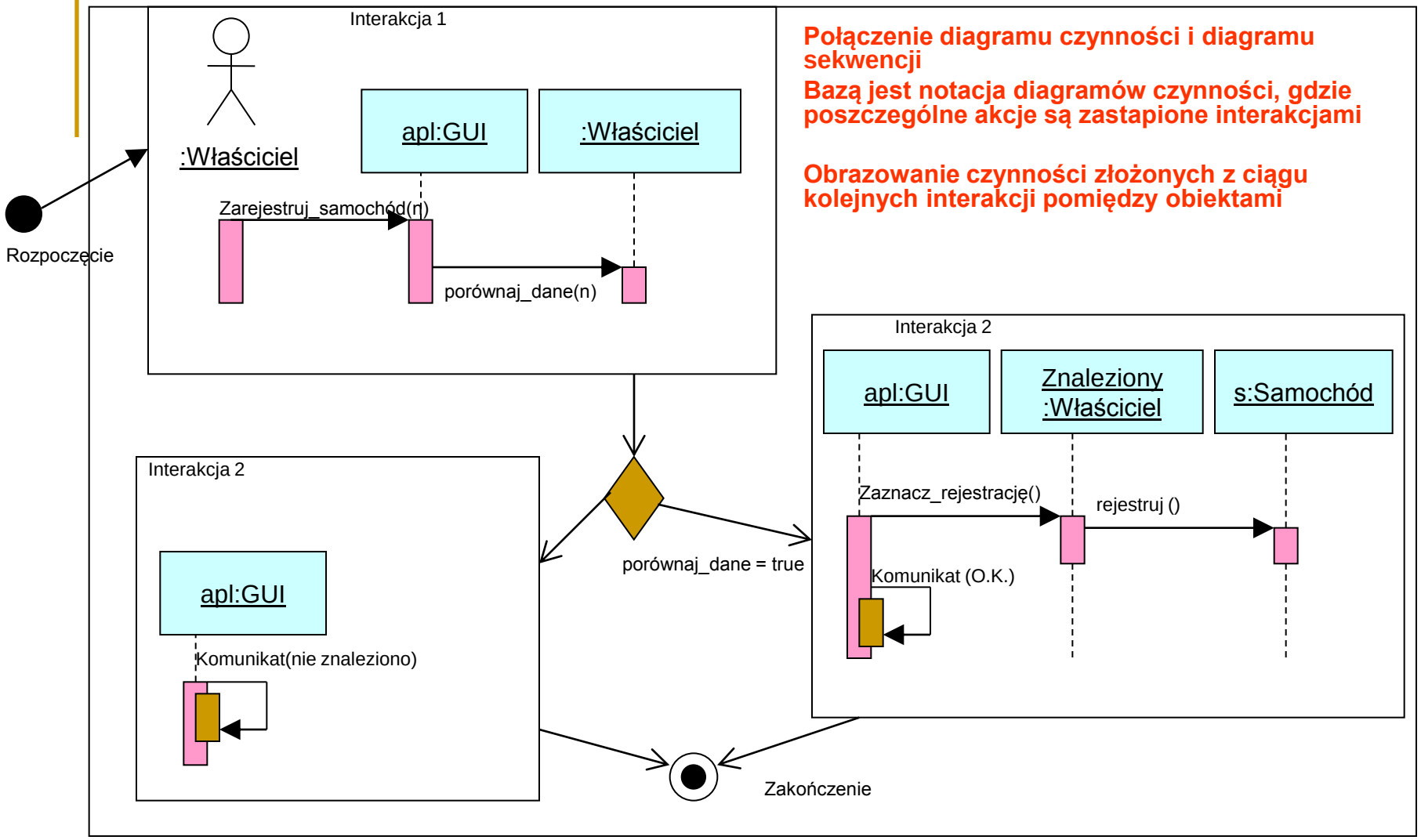
→ Aktorzy również włączeni jako obiekty. Nazwa komunikatu nad strzałkami obrazującymi ich wymianę. Komunikat powrotny – powrót sterowania

MODELOWANIE DYNAMIKI
diagram interakcji (komunikacji)

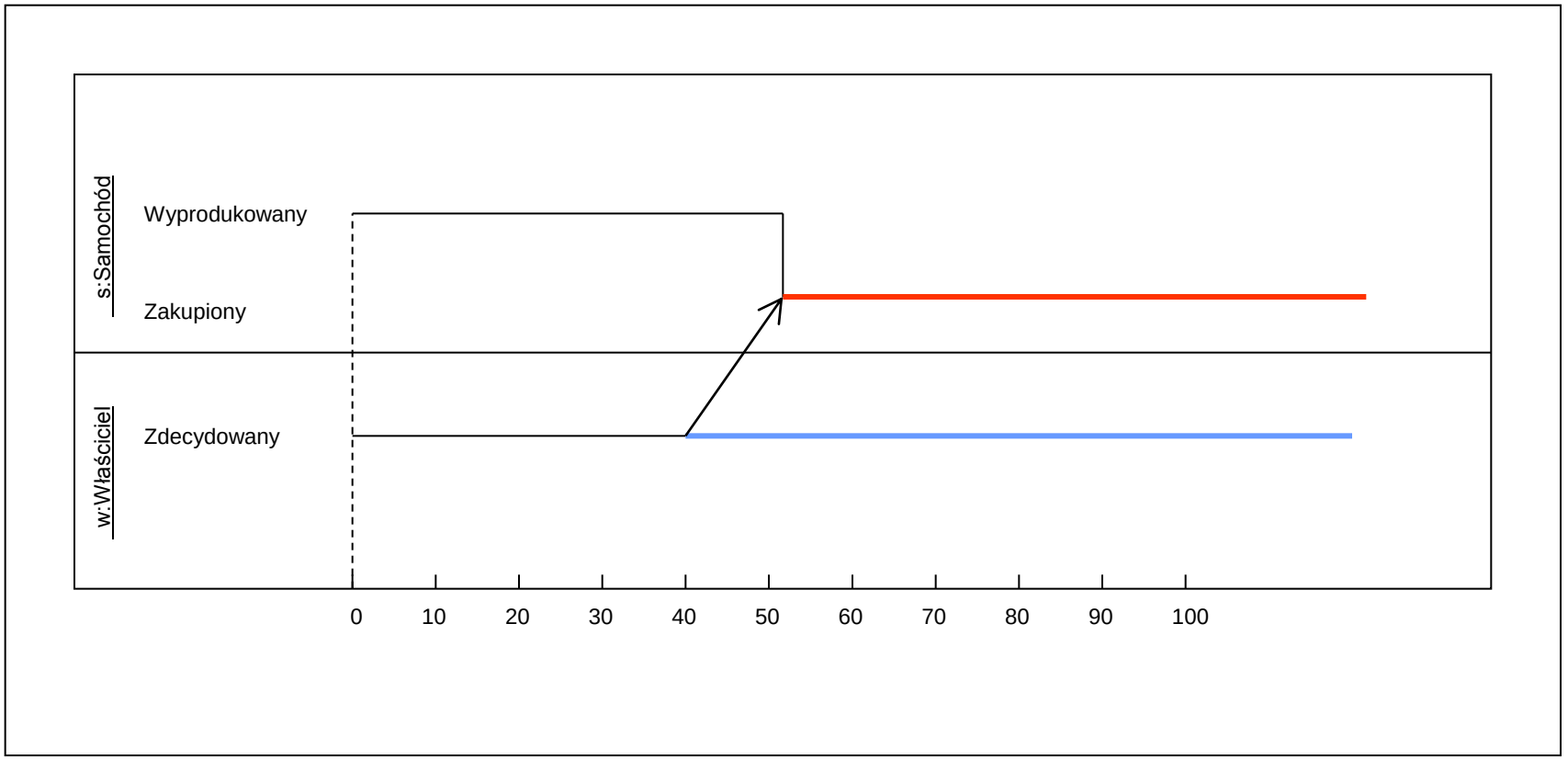


→ **Uboższa notacja niż w diagramach sekwencji**
→ **Komunikaty oznaczone kolejnymi cyframi**
→ **Brak komunikatów zwrotnych**
→ **Aktorzy również włączeni jako obiekty. Brak jednoznacznych pętli**

MODELOWANIE DYNAMIKI
diagram interakcji (opisu interakcji)



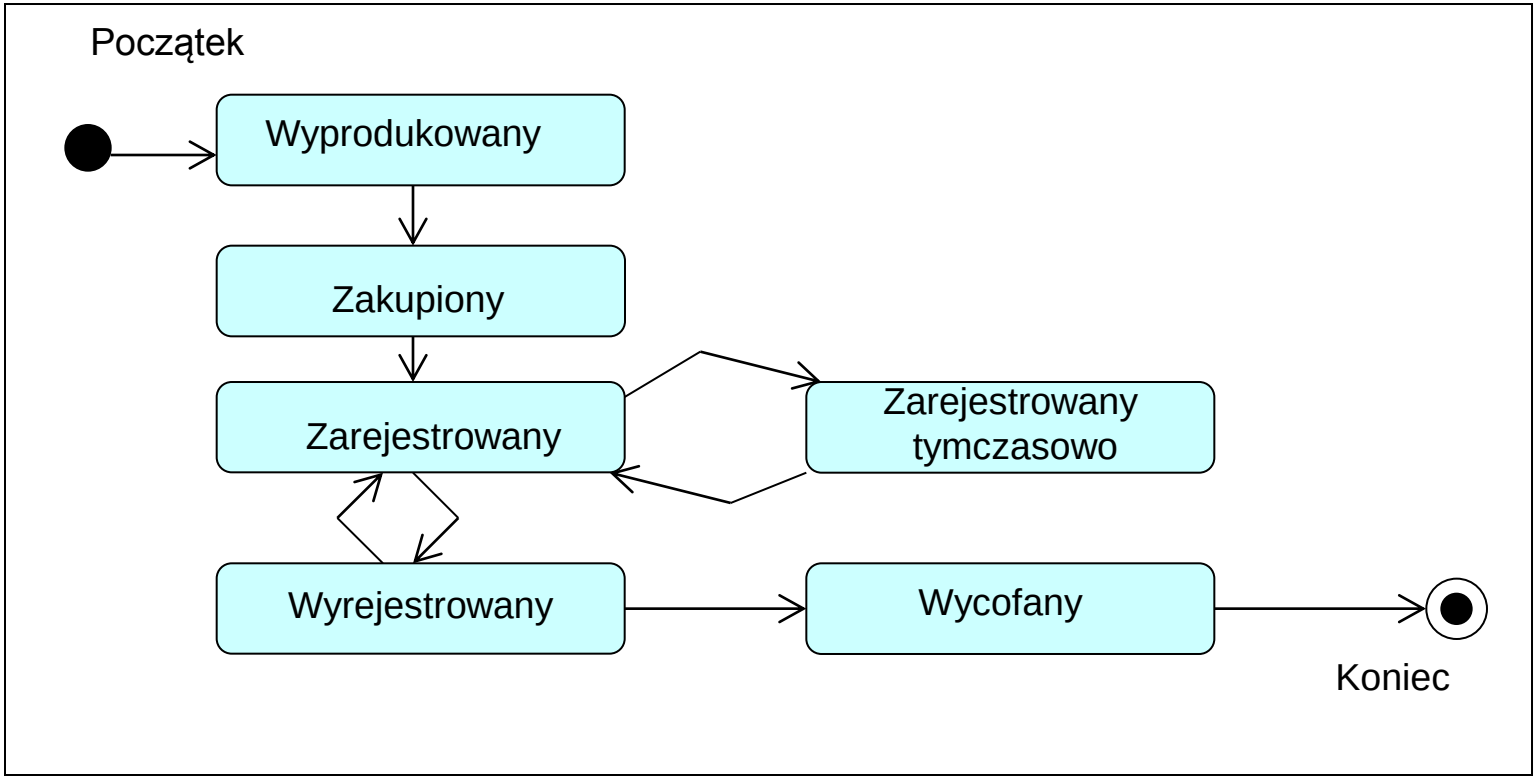
MODELOWANIE DYNAMIKI
diagram interakcji (następstw)



→ **Pokazują następstwo zdarzeń w sekwencji stanu obiektów oraz zmiany tych stanów pod wpływem komunikatów przesyłanych między obiektami.**

→ **Przestawiają zależności czasowe warunkujace przesyłanie komunikatów**

MODELOWANIE DYNAMIKI
diagram maszyny stanów



→ **określają sposób zachowania się wybranego elementu modelu (obiektu) w postaci przejść pomiędzy jego stanami.**

→ **Stan – pewna, stabilna sytuacja w jakiej znajduje się dany element**

→ **Przejścia do innych stanów mogą następować pod wpływem określonych zdarzeń (np. Komunikatów otrzymanych od innych elementów modelu)**

MODELOWANIE OBIEKTOWE

Praktyczne wykorzystanie diagramów struktury i dynamiki

→ **Użytkownik lub zamawiający system:**
diagramy przypadków użycia, diagramy klas, diagramy czynności

→ **Architekci systemu oprogramowania**

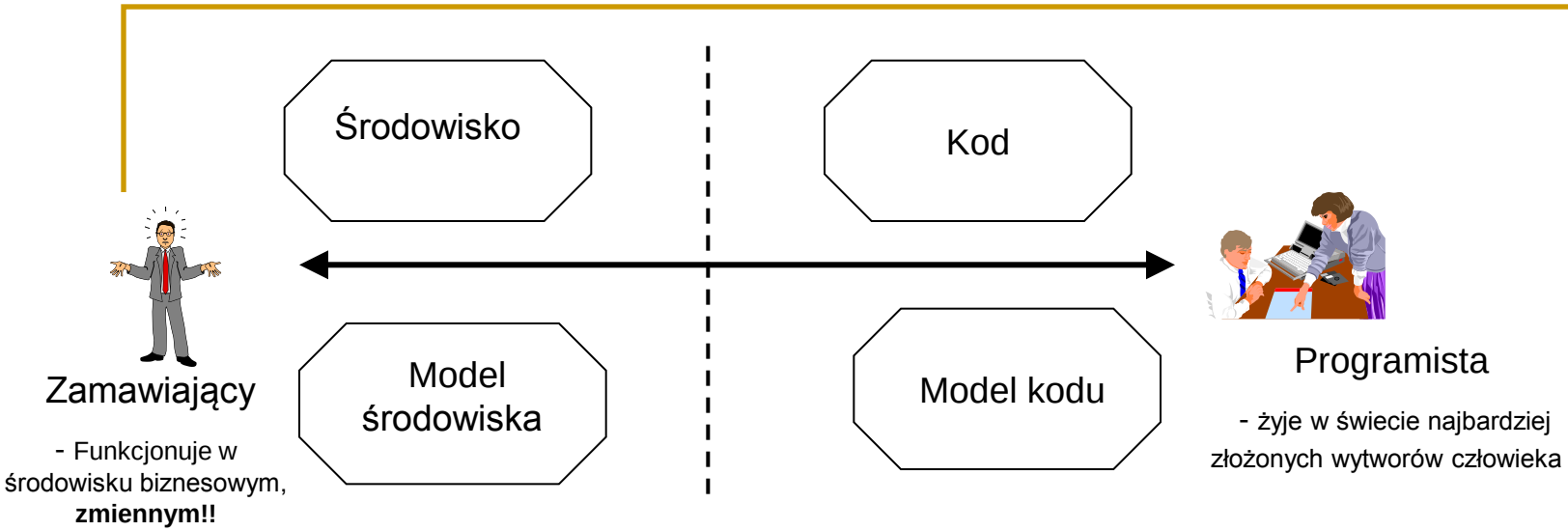
dodatkowo – diagramy komponentów, diagramy wdrożeń, diagramy sekwencji, diagramy składowych

→ **Projektanci systemów czasu rzeczywistego**

diagramy następstw, diagramy maszyny stanów

Od Środowiska do Kodu

Obszary aktywności



→ **Modele obiektowe – próba ustanowienia wspólnego języka dwóch światów zamawiającego i programisty (światy bardzo złożone)**

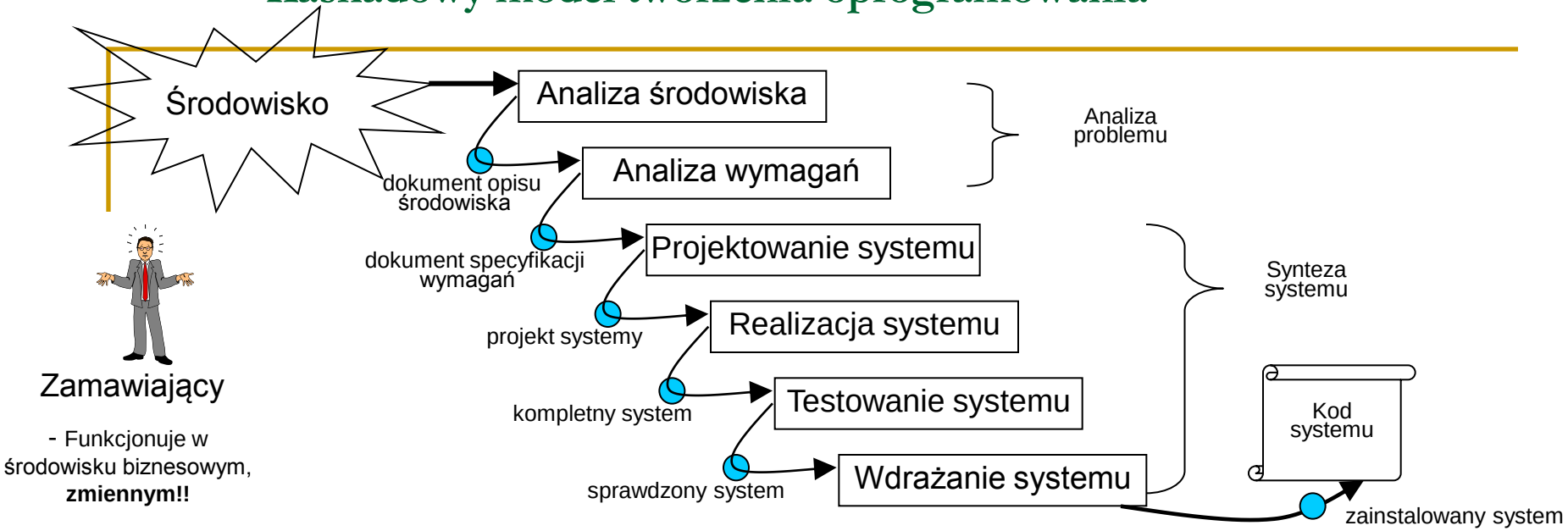
→ **Tłumaczenie modelu środowiska na model kodu - transformacje**

→ **Modelarze transformacji środowiska:**

Zamawiający – sprawdzenie zgodności ze środowiskiem
Analityk środowiska – tworzenie opisu środowiska
Użytkownik – sprawdzanie wymagań
Analityk wymagań – tworzenie wymagań
Architekt – projektowanie systemu
Projektant – projektowanie podsystemów
Programista – wykonanie systemu
Recenzent – sprawdzenie jakości produktów
Tester – sprawdzenie jakości kodu

Od Opisu Środowiska do Kodu

Kaskadowy model tworzenia oprogramowania



→ **Model kaskadowy (waterfall) – proces wytwórczy oprogramowania składający się z faz**

→ **Fazy – etapy aktywności twórczej zakończone określonym produktem składającym się na tzw. kamienie milowe (milestone) – określają cele poszczególnych faz**

→ **Faza analizy i syntezy**

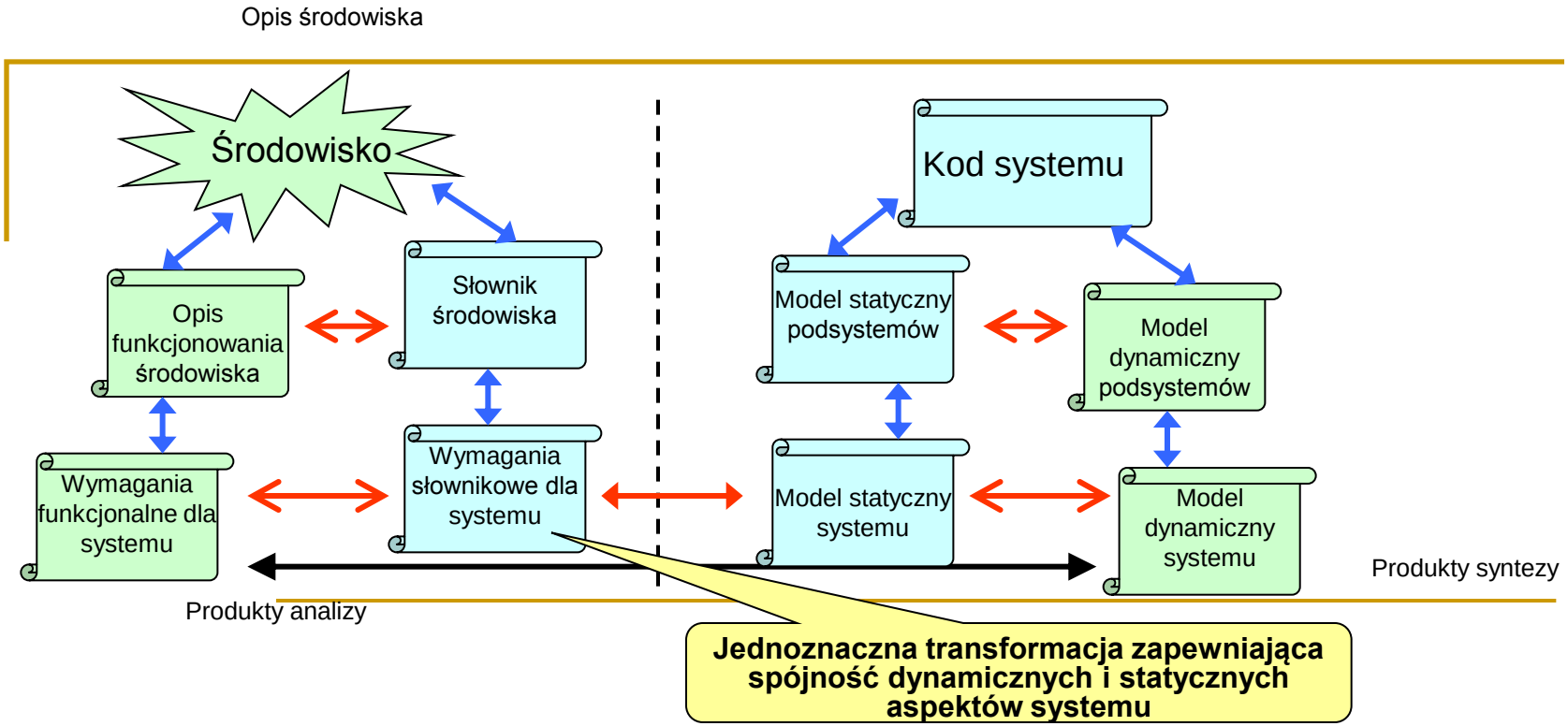
→ **Wady** (większość tak realizowanych projektów kończy się fiaskiem):

Brak iteracji na poszczególnych etapach

Brak jednoznacznych procedur przekształcenia wymagań środowiska w system oprogramowania

Brak ciągu modeli wskazujących na sposób transformacji od środowiska do kodu.

CIĄG MODELI Obszary aktywności



→ **Wszystkie modele są połączone transformacjami.**

→ **Transformacja to przekształcenie modelu na inny z zachowaniem informacji, który element modelu jest przekształcany i w jaki sposób.**

→ **Transformacje poziome są realizowane w ramach jednej „roli" w projekcie, na jednym poziomie kaskady, pionowe pozwalają na przejście pomiędzy kaskadami.**

Słownik środowiska – model statyczny
Opis funkcjonowania środowiska – model dynamiczny

→ Brak modeli służących testowniu systemów

MODELOWANIE OBIEKTOWE

Tworzenie systemów sterowane modelami

→ **OMG (2000)** *(Model Driven Architecture - **MDA**)* związane z Tworzeniem programów sterownych modelami *(Model driven Developmnet)*

idea wytwarzania oprogramowania oparta na modelach i ich przekształceniach

→ **Podstawowe element MDA:**

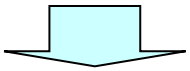
definicja modelu na danym etapie cyklu
definicja przekształcenia modelu w inny model

→ **Pierwotna idea MDA – dwa poziomy modelowania:**

poziom niezależny od platformy (PIM – *Platform Independent Model*) – specyfikacja wymagań
poziom zależny od platformy (PSM –*Platform Specyfic Model*) – realizacja wymagań

→ **Wyrażenie modeli i transformacji pomiędzy nimi z pomocą UML:**

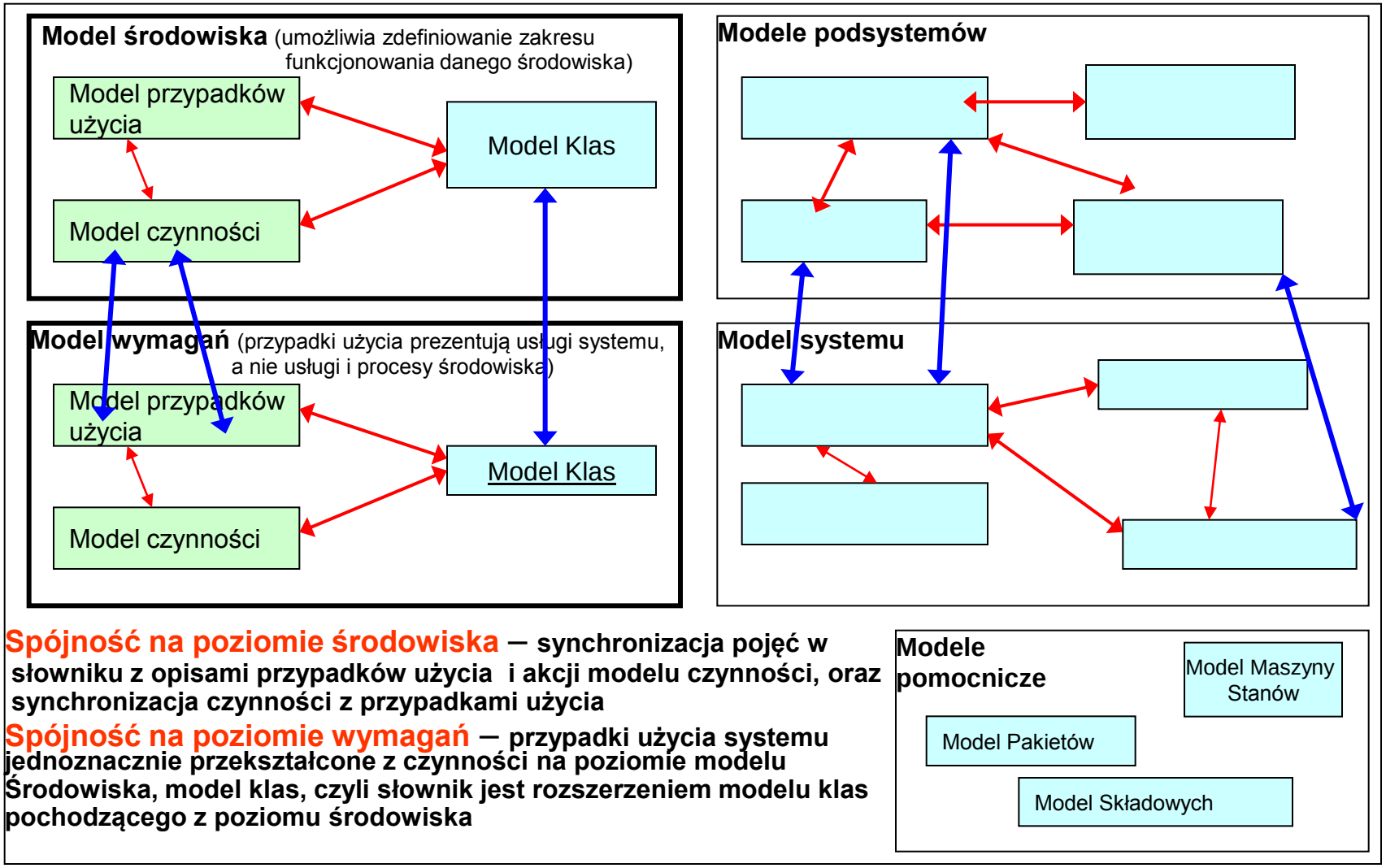
1. Które modele modele są potrzebne na poszczególnych etapach tworzenia oprogramowania?
2. Jakie warunki powinny spełniać te modele na różnych etapach?
3. Jakie przekształcenia modeli są potrzebne?
4. Które elementy modeli powinny być nawzajem przekształcane i w jaki sposób?



Sposób określenia ścieżki od wymagań do kodu!!

Tworzenie systemów sterowane modelami

Model środowiska

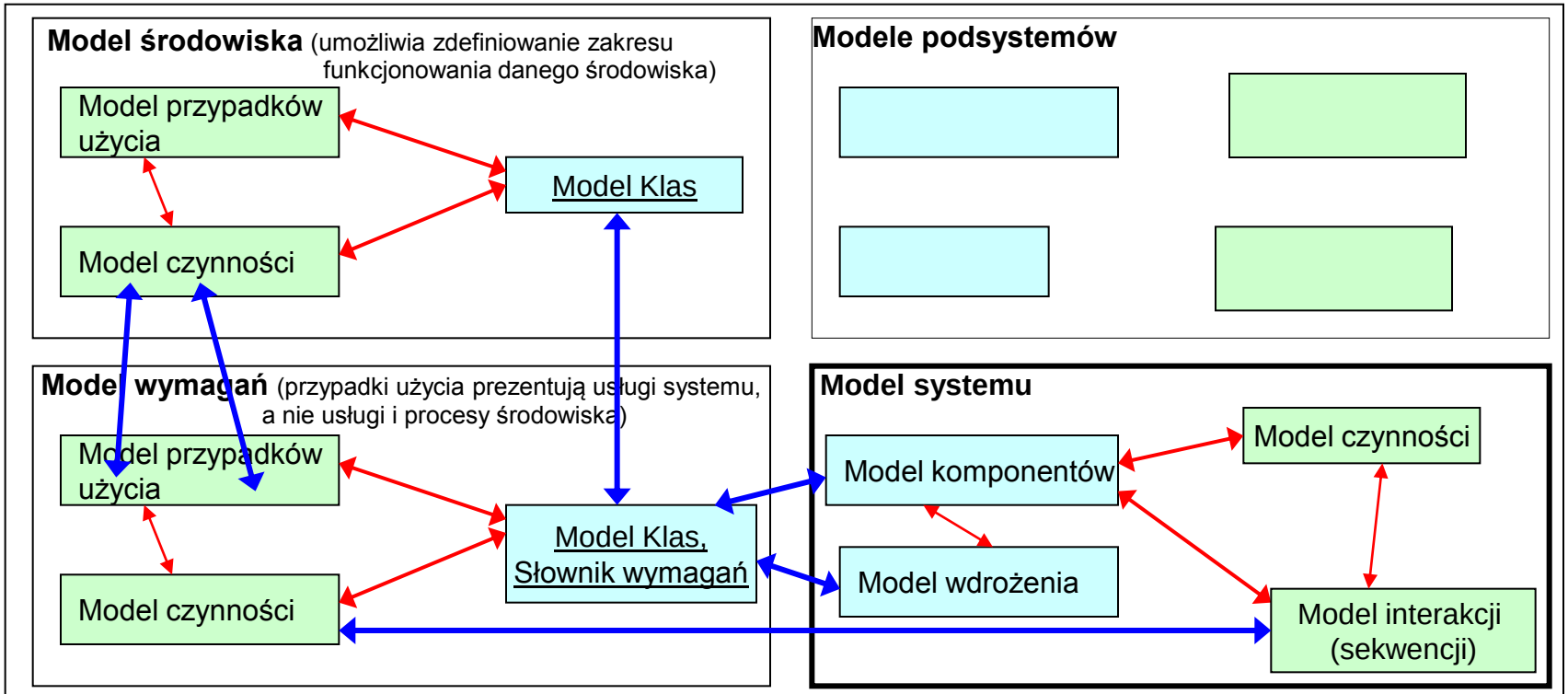


Spójność na poziomie środowiska – **synchronizacja pojęć w słowniku z opisami przypadków użycia i akcji modelu czynności, oraz synchronizacja czynności z przypadkami użycia**

Spójność na poziomie wymagań – **przypadki użycia systemu jednoznacznie przekształcone z czynności na poziomie modelu Środowiska, model klas, czyli słownik jest rozszerzeniem modelu klas pochodzącego z poziomu środowiska**

Tworzenie systemów sterowane modelami

Przekształcenie modelu wymagań w model systemu

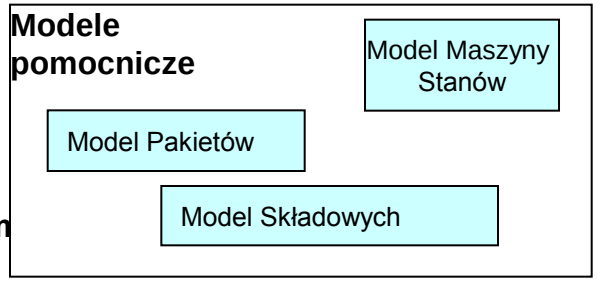


Model komponentów – **podział systemu na podsystemy (model logiczny), specyfikacja interfejsów do komunikacji pomiędzy podsystemami.**

Model wdrożeń – **podział systemu na elementy fizyczne (maszyny mające pamięć i możliwości przetwarzania.**

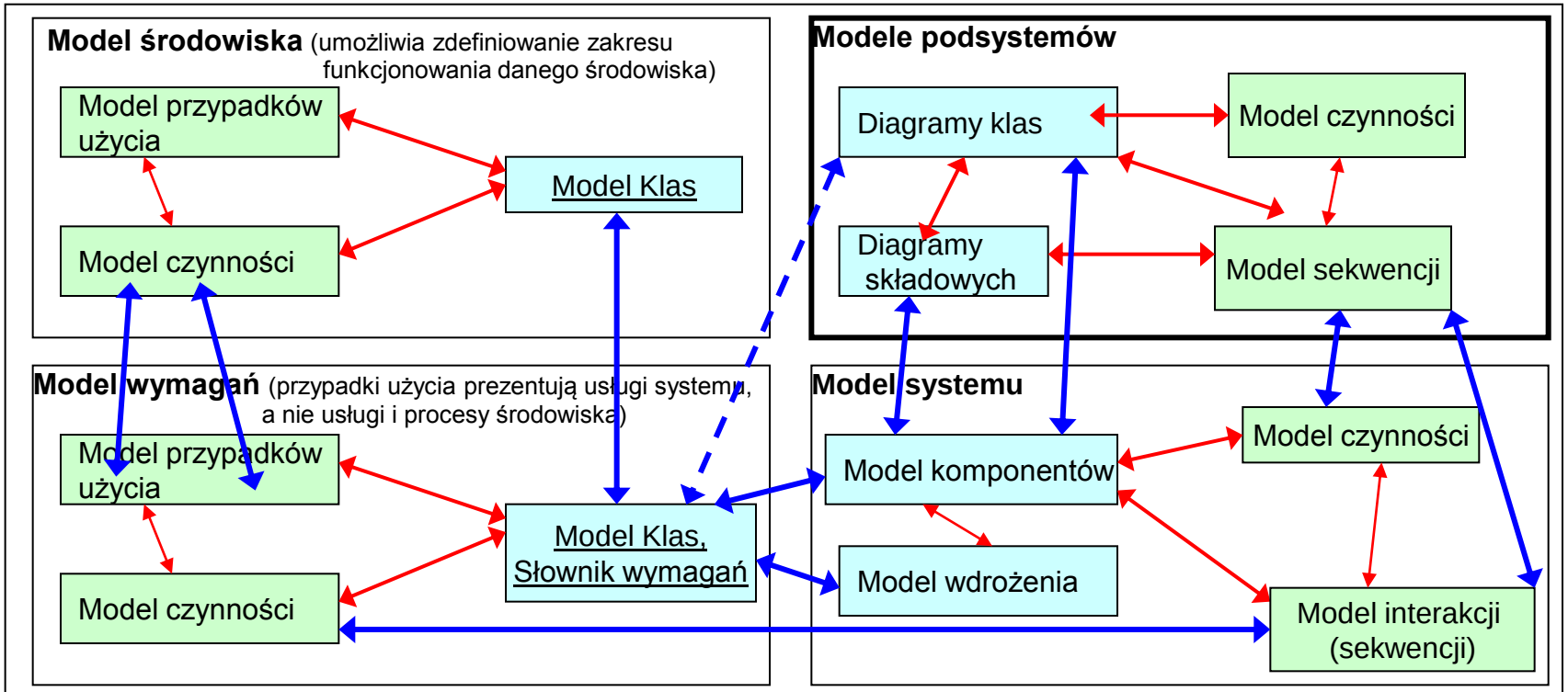
Diagramy czynności – **pozwalają zaprojektować operacje dostarczane przez interfejsy – spójność – przyporządkowane operacjom poszczególnych interfejsów.**

Diagramy interakcji– **realizacja poszczególnych funkcjonalności systemu. Interakcje pomiędzy komponentami a interfejsami pokazują działanie systemu – spójność – interakcja pomiędzy interf. a komp.**



Tworzenie systemów sterowane modelami

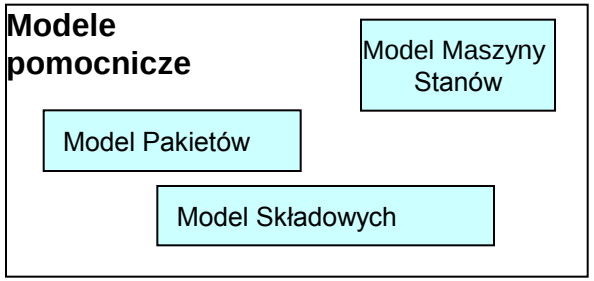
Uszczegółowienie systemu



Diagramy składowych, diagramy klas – **uszczegółowienie podsystemów wynikające z modelu komponentów (związek z modelem klas modelu wymagań)**

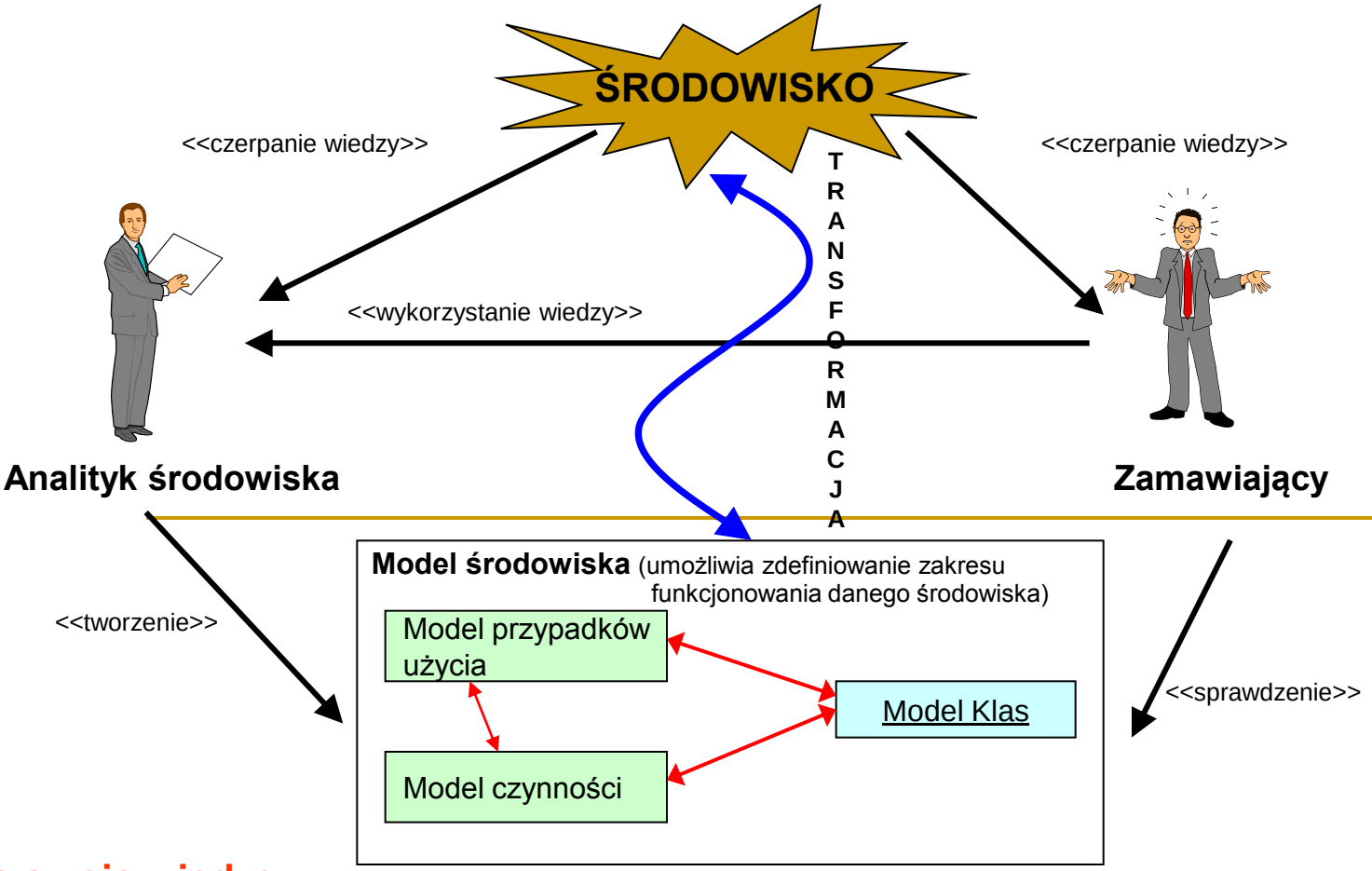
Model czynności i sekwencji– **zależność od modelu klas podobna do zależności w modelu systemu od modelu komponentów**

Diagramy interakcji– **realizacja poszczególnych funkcjonalności systemu. Interakcje pomiędzy komponentami a interfejsami pokazują działanie systemu – spójność – interakcja pomiędzy interf. a komp.**



FORMUŁOWANIE WYMAGAŃ

Transformacja środowiska w model środowiska



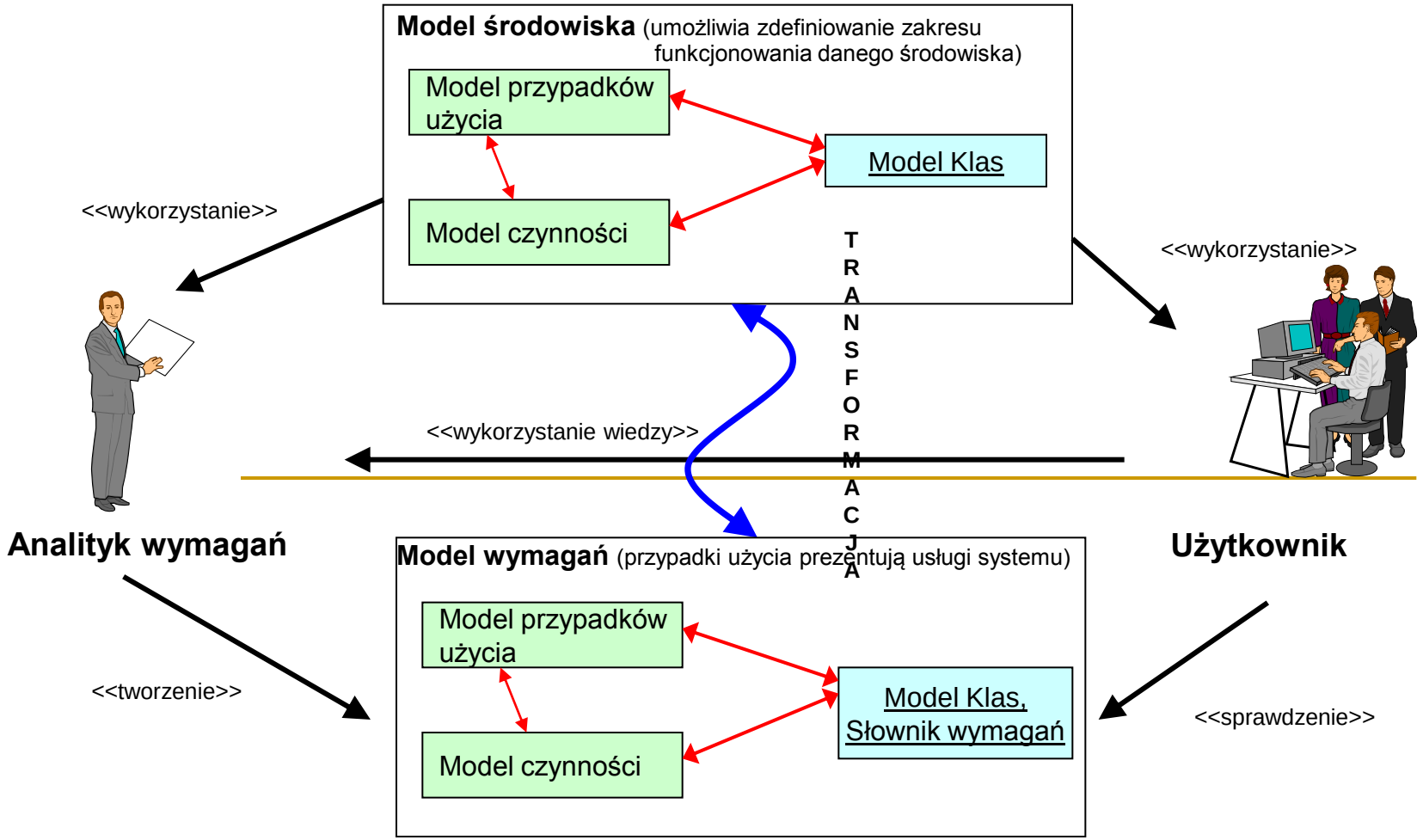
→ **Każdy ma swoją wiedzę.**

→ **Zamawiający ma cele i priorytety nie do zautomatyzowania**

→ **Model analityczny środowiska – uzyskany z transformacji środowiska.** (**Odwrócenie – model zmieniający środowisko jest projektem środowiska – środowisko sterowne modelami)**

FORMUŁOWANIE WYMAGAŃ

Transformacja modelu środowiska w model wymagań

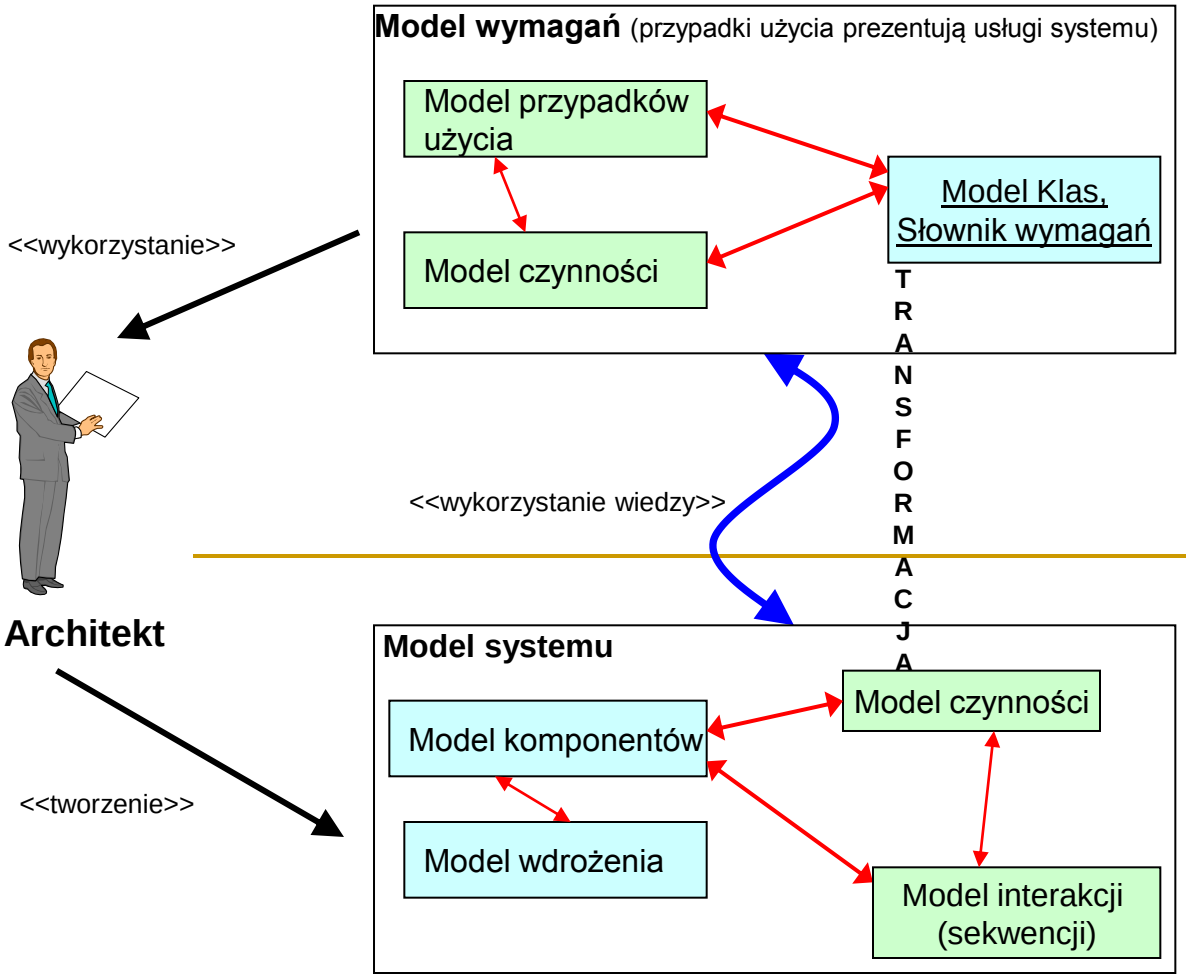


→ **Stabilny model środowiska daje podstawę dla stworzenia modelu wymagań dla odpowiedniego systemu oprogramowania – należy uwzględnić użytkownika**

→ **Dopuszczalne – na bazie wymagań powstaje model środowiska**

REALIZACJA WYMAGAŃ

Transformacja modelu wymagań w model systemu



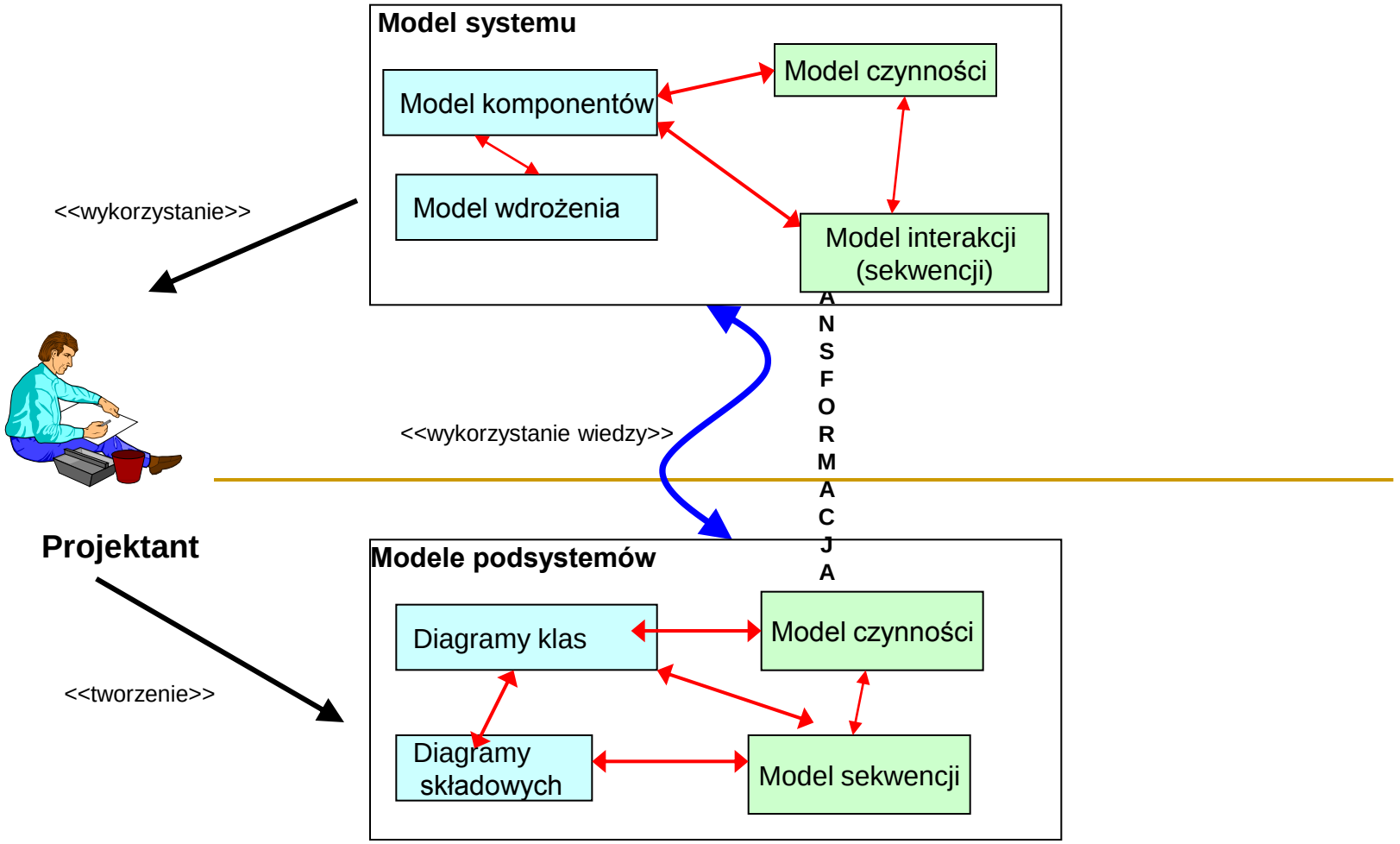
→ **Podział systemu na logiczne jednostki funkcjonalne – komponenty**

→ **Dbałość o spójność podziału na komponenty i komunikacji za pomocą interfejsów**

→ **Komponenty grupują elementy realizujące zwarty podsystem** (duża zwartość komponentów oraz słabe więzy z innymi komponentami)

REALIZACJA WYMAGAŃ

Transformacja modelu systemu w model podsystemu

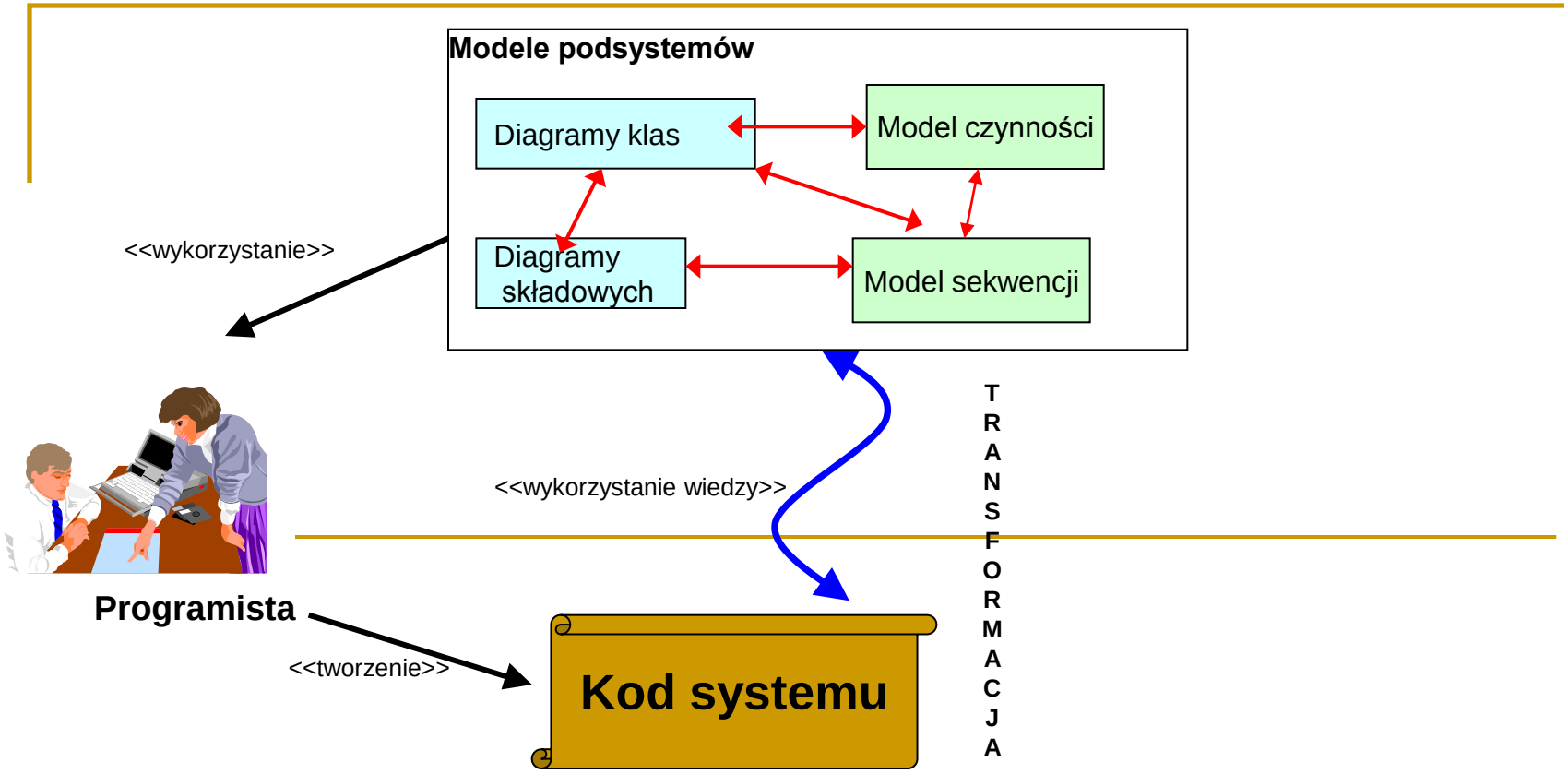


→ **Projektowanie struktury i dynamiki komponentów**

→ **Dbałość o jakość realizacji zawartości komponentów (wraz z „programistami”)**

REALIZACJA WYMAGAŃ

Transformacja modelu podsystemu do kodu



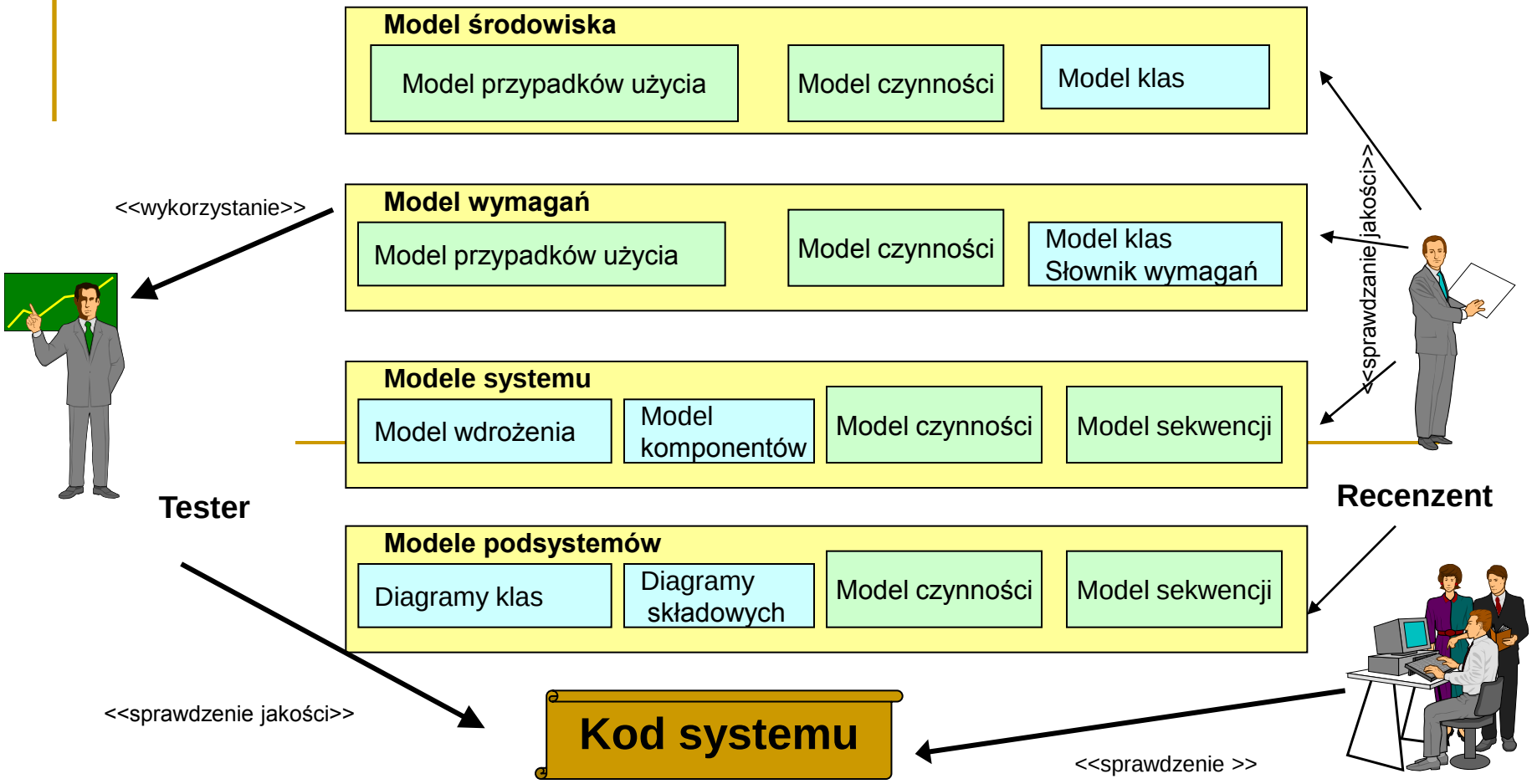
→ Decyzje projektowe zawarte w modelu w modelu podsystemu są realizowane przez programistę. Wykorzystując składnię określonego języka zapisuje strukturę i algorytmy przetwarzania zaprojektowane przez projektanta, proste.

→ W przypadku braku narzędzi do automatycznego generowania kodu zadanie żmudne.

→ Generatory kodu budujące szkielet kodu na podstawie modelu klas. Dopuszcza się transformacje odwrotne – łączenie funkcji programisty i projektanta

KONTROLA JAKOŚCI

Zapewnienie jakości w projekcie tworzenia oprogramowania



→ **Wpływ ponownego wykorzystania na jakość projektu**

→ **Tworzenie modeli projektowych (podsystemów) wzorce projektowe**

→ **Szkielety architektoniczne – wzorce, modele wymagań - wzorce.**