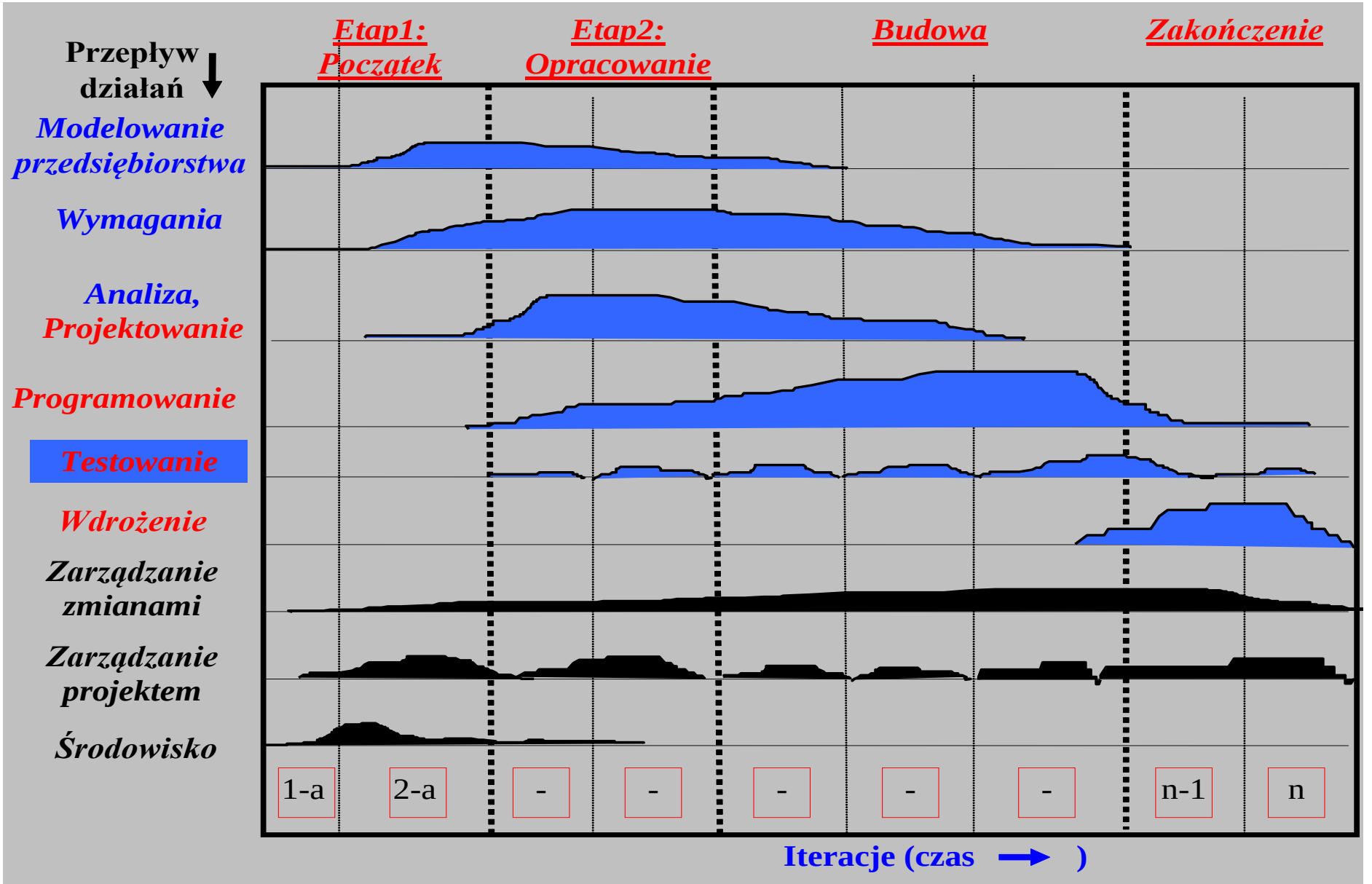
Iteracyjno-rozwojowy proces tworzenia oprogramowania Wykład 3 – część 1

Zofia Kruczkiewicz

Zunifikowany iteracyjno- przyrostowy proces tworzenia oprogramowania – kiedy?



Przepływy czynności

- Modelowanie przedsiębiorstwa – opis dynamiki i struktury przedsiębiorstwa
- Wymagania – zapisanie wymagań metodą opartą na przypadkach użycia
- Analiza i projektowanie – zapisanie różnych perspektyw architektonicznych
- Implementacja – tworzenie oprogramowania, testowanie modułów, scalanie systemu
- Testowanie – opisanie danych testowych, procedur i metryk poprawności
- Wdrożenie – ustalenie konfiguracji gotowego systemu
- Zarządzanie zmianami – panowanie nad zmianami i dbanie o spójność elementów systemu
- Zarządzanie projektem - opisane różnych strategii prowadzenia procesu iteracyjnego
- Określenie środowiska – opisanie struktury niezbędnej do opracowania systemu

Co i jak wykonać?

Perspektywy projektowania obiektowych systemów informacyjnych [4]

- **koncepcji** (model analizy)

  (co obiekty powinny robić?)

- **specyfikacji interfejsów** (model projektowy)

  (jak używać obiektów?)

- **implementacji** (implementacja)

  (w jaki sposób zaimplementować interfejs ?)

- **tworzenia i zarządzania obiektami** (implementacja)

  (*obiekt A w roli fabryki obiektów tworzy obiekt B i/lub zarządza obiektem)*

- **używania obiektów** (implementacja)

  (*obiekt A tylko używa obiektu B – nie może go jednocześnie tworzyć;*

  opiera się na hermetyzacji i polimorfiźmie obiektu B)

Perspektywy rozumienia obiektów – identyfikacji obiektów [4]

- **Perspektywa koncepcji** (modelu konceptualnego)

  - obiekt jest zbiorem różnego rodzaju odpowiedzialności

- **Perspektywa specyfikacji** (modelu projektowego)

  - obiekt jest zbiorem metod (zachowań), które mogą być wywoływane przez metody tego obiektu lub innych obiektów

- **Perspektywa implementacji** (kodu źródłowego)

  - obiekt składa się z kodu metod i danych oraz interakcji między nimi

Perspektywy skalowania systemu – tworzenia, zarządzania i używania obiektów [4]

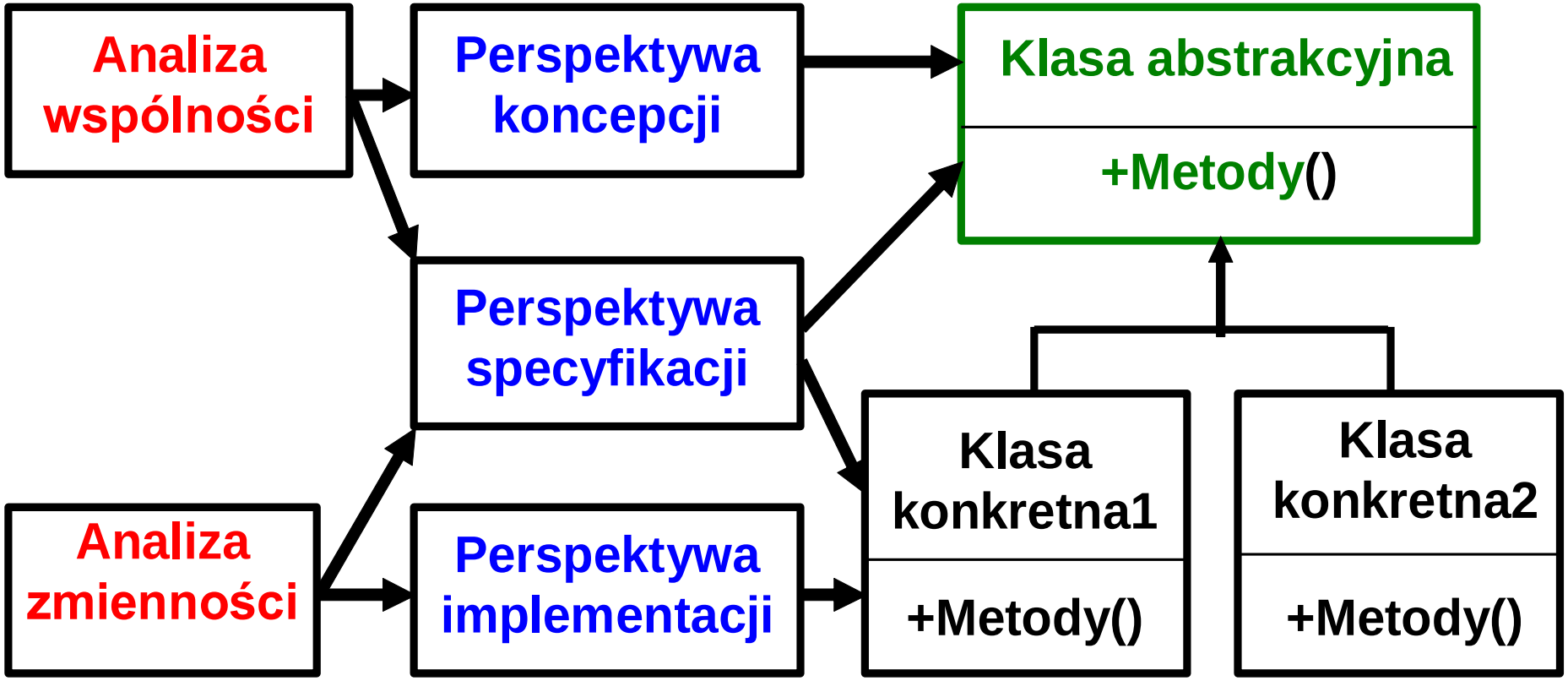
- **Perspektywa tworzenia i zarządzania obiektami**

  Zmiany w implementacji obiektów dotyczą obiektów czyli fabryk obiektów (tworzących te obiekty i zarządzających tworzeniem tych obiektów)

- **Używanie obiektów**

  Zmiana implementacji obiektów nie zmienia implementacji obiektów, które używają zmieniane obiekty

Metoda identyfikacji obiektów i klas [4]



Związek między perspektywą specyfikacji, koncepcji i implementacji

Zależności między analizą, projektowaniem i implementacją [4]

Związek między perspektywą koncepcji i specyfikacji

- Perspektywa specyfikacji określa interfejs potrzebny do obsługi wszystkich przypadków danego problemu (czyli **część wspólną** określoną przez perspektywę koncepcji)

Związek pomiędzy perspektywą specyfikacji i implementacji

- Biorąc pod uwagę określoną specyfikację ustala się, w jaki sposób należy zaimplementować poszczególne przypadki (czyli **część zmienną**)

Produkt – diagramy UML wspierające zunifikowany iteracyjno - przyrostowy proces tworzenia oprogramowania [3]

Diagramy modelowania struktury

1.1. Diagramy pakietów

1.2. Diagramy klas

1.3. Diagramy obiektów

1.4. Diagramy mieszane

1.5. Diagramy komponentów

1.6. Diagramy wdrożenia

Diagramy UML modelowania zachowania

2.1. Diagramy przypadków użycia

2.2. Diagramy aktywności

2.3. Diagramy stanów

2.4. Diagramy komunikacji

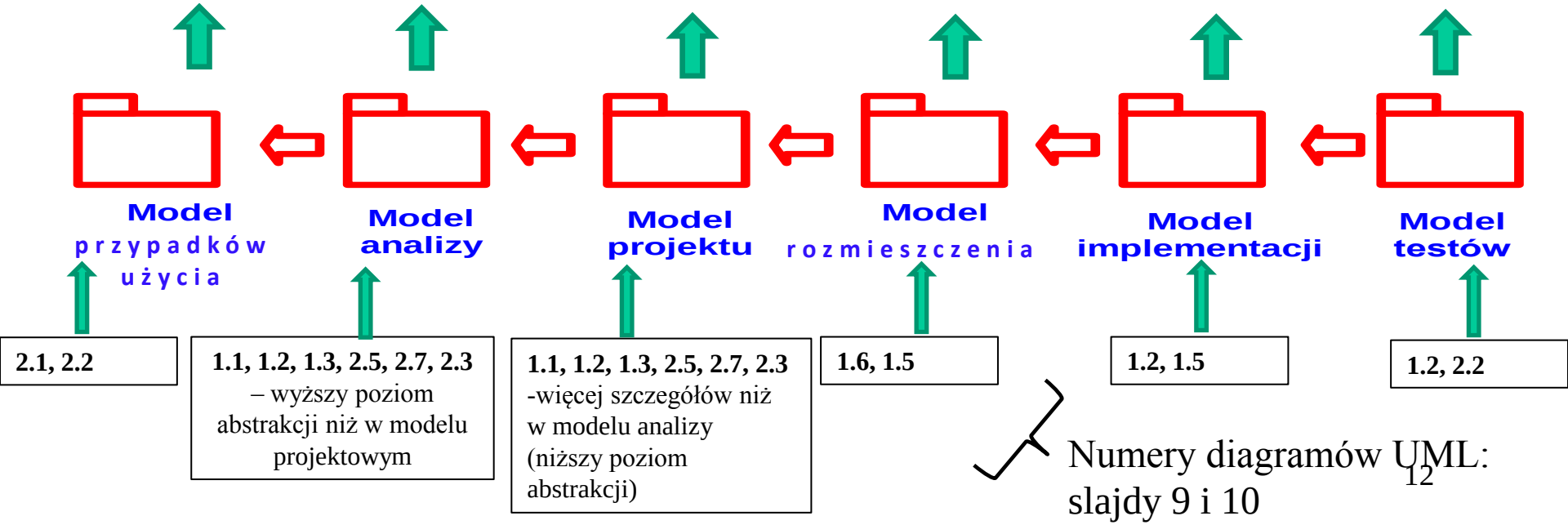
2.5. Diagramy sekwencji

2.6. Diagramy czasu

2.7. Diagramy interakcji

MDA (Model Driven Architecture) – przekształcanie modeli produktu w procesie tworzenia oprogramowania

Modelowanie struktury i dynamiki systemu	Implementacja systemu,	struktury i dynamiki generowanie kodu
Perspektywa koncepcji ***co należy wykonać?***	Perspektywa specyfikacji ***jak należy używać?***	Perspektywa implementacji ***jak należy wykonać?***
• model problemu np. przedsiębiorstwa • wymagania **(model przypadków użycia)** • analiza **(model analizy)** • testy modeli	• projektowanie **(model projektu, model rozmieszczenia):** architektura sprzętu i oprogramowania; dostęp użytkownika; przechowywanie danych) • testy modeli	• programowanie **(model implementacji)** (specyfikacja programu : deklaracje, definicje; dodatkowe struktury danych: struktury „pojemnikowe”, pliki, bazy danych) • testy oprogramowania **(model testów)** • wdrażanie • testy wdrażania



Numery diagramów UML: slajdy 9 i 10