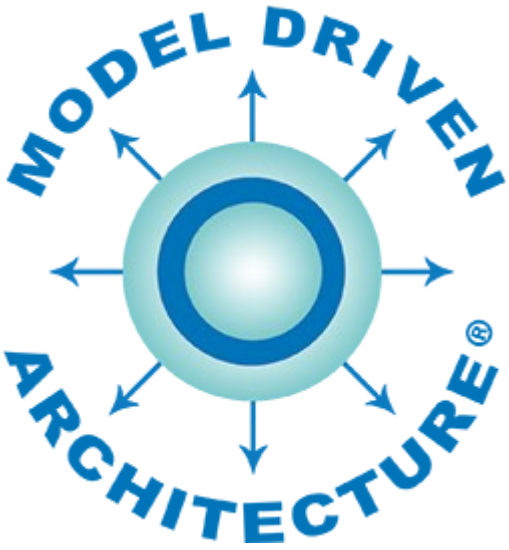


Architektura Sterowana Modelem

Model Driven Architecture

prezentacja 2

Projektowanie systemu informatycznego metodą MDA

wersja 1.0

dr inż. Paweł Głuchowski

Wydział Informatyki i Telekomunikacji, Politechnika Wrocławska

Treść prezentacji

1. System informatyczny
2. Inżynieria oprogramowania
3. Cykl życia oprogramowania
4. Architektura Sterowana Modelem
5. Poziomy abstrakcji modelu
6. Etapy MDA

1

System informatyczny

System Informatyczny (SI)

- **System wspierający działanie użytkownika (organizacji) w realizacji biznesowych zadań** przez:
  - przechowywanie, przesyłanie, zarządzanie i przetwarzanie danych (artefakty, np. informacje);
  - zarządzanie i wykonywanie funkcji i procesów organizacyjnych (biznesowych) użytkownika (organizacji);
  - w celu spełnienia/spełniania określonych wymagań.

- Składa się z powiązanych ze sobą:
  - elementów nieformalnych (użytkownicy i ich role);
  - elementów formalnych (procedury organizacyjne, kontrola bezpieczeństwa w sensie security, bazy wiedzy, tutoriale itp.);
  - elementów technicznych (sprzęt, oprogramowanie systemowe, bazy danych itp.).

Cel modelowania i analizowania SI

- Zbudować / przebudować / rozbudować SI:
  - konkretny SI,
  - dla konkretnego użytkownika (organizacji),
  - realizujący konkretne zadania.

2

Inżynieria oprogramowania

Inżynieria oprogramowania (IO)

- **Wiedza i praktyka techniczna:**
  - dotycząca wszystkich faz cyklu życia oprogramowania;
  - aby uzyskać produkt (np. SI) wysokiej jakości (tańszy, mniej zawodny, efektywniej działający);
  - systemowe podejście z ilościową oceną.

Na IO składają się

- **Zarządzanie projektem informatycznym** – organizacja i kierowanie realizacją projektu (np. mającego wykonać i wdrożyć SI).
- **Określenie środowiska programistycznego** – wybór procedur organizacyjnych, sprzętu i oprogramowania dla realizacji projektu informatycznego.
- **Analiza i model SI:**
  - model odbiorcy SI – opis struktury i dynamiki jego organizacji / przedsiębiorstwa,
  - analiza wymagań stawianych SI (w tym analiza przypadków użycia),
  - model struktury i zachowania SI (w tym model oprogramowania systemu i jego wdrożenia).

Na IO składają się

- **Implementacja SI** – stworzenie oprogramowania (w tym kodu).
- **Testowanie SI** – testy jednostkowe, funkcjonalne, integracyjne, akceptacyjne i inne:
  - na etapie implementacji i potem,
  - przygotowanie danych testowych,
  - wybór procedur i metryk poprawności.
- **Wdrożenie SI** – konfiguracja wykonanego oprogramowania i jego uruchomienie u odbiorcy.
- **Konserwacja SI** – zarządzanie zmianami, dbanie o spójność elementów, poprawa bezpieczeństwa.

3

Cykl życia oprogramowania

Cykl życia (wytwarzania) oprogramowania SI

1. **Modelowanie struktury i dynamiki systemu**
   – perspektywa koncepcji.

- Określa, **co** trzeba wykonać jako SI.
- Zawiera:
  - model odbiorcy,
  - analizę wymagań,
  - koncepcję testów.

Cykl życia (wytwarzania) oprogramowania SI

2. **Implementacja struktury i dynamiki systemu i wytworzenie kodu**
   – perspektywa specyfikacji.

- Określa, **jak** trzeba będzie **używać** SI.
- Zawiera:
  - model projektu – struktura i zachowanie SI,
    w tym: architektura sprzętu i oprogramowania, dostęp użytkownika, przechowywanie danych itp.;
  - testy funkcjonalne i akceptacyjne projektu.

Cykl życia (wytwarzania) oprogramowania SI

3. **Implementacja struktury i dynamiki systemu i wytworzenie kodu** – perspektywa implementacji.

- Określa, **jak** trzeba będzie **wykonać** SI.
- Zawiera:
  - oprogramowanie SI – uzupełnienie modelu SI o dodatkowe deklaracje, definicje, struktury danych, bazy danych itp.;
  - testy oprogramowania (w tym: jednostkowe, integracyjne);
  - wdrożenie SI;
  - testy systemu (w tym: akceptacyjne, wdrożeniowe).

Model kaskadowy

- Liniowa, sekwencyjna kolejność realizacji etapów:

  określenie wymagań → projektowanie → implementacja → testowanie → wdrożenie → konserwacja

- **Zalety modelu:**
  - proste zależności między jego etapami,
  - łatwo nim zarządzać.
- **Wady modelu:**
  - nie można cofnąć się do etapu wymagającego zmian,
  - trudno usunąć błędy z początkowych etapów,
  - powstałe oprogramowanie może nie spełnić wymagań.

Model kaskadowy typu V

- Liniowa, sekwencyjna kolejność realizacji etapów,
  gdzie rozwojowi oprogramowania towarzyszy testowanie:

  określenie wymagań (i testów akceptacyjnych)
  → projektowanie systemu (i jego testów integracyjnych)
  → projektowanie komponentów systemu (i ich testów integracyjnych)
  → implementacja → testy jednostkowe → testy integracyjne komponentów → testy integracyjne systemu → testy akceptacyjne
  → wdrożenie → konserwacja

- **Zalety modelu:**
  - weryfikacja i walidacja planowana jest na każdym jego etapie,
  - błędy z początkowych etapów są mniejsze.
- **Wady modelu:**
  - nie można cofnąć się do etapu wymagającego zmian,
  - trudno usunąć błędy z początkowych etapów.

Model przyrostowy

- Ewolucyjny model kaskadowo-iteracyjny.
- Każda iteracja dotyczy innego komponentu składowego SI:

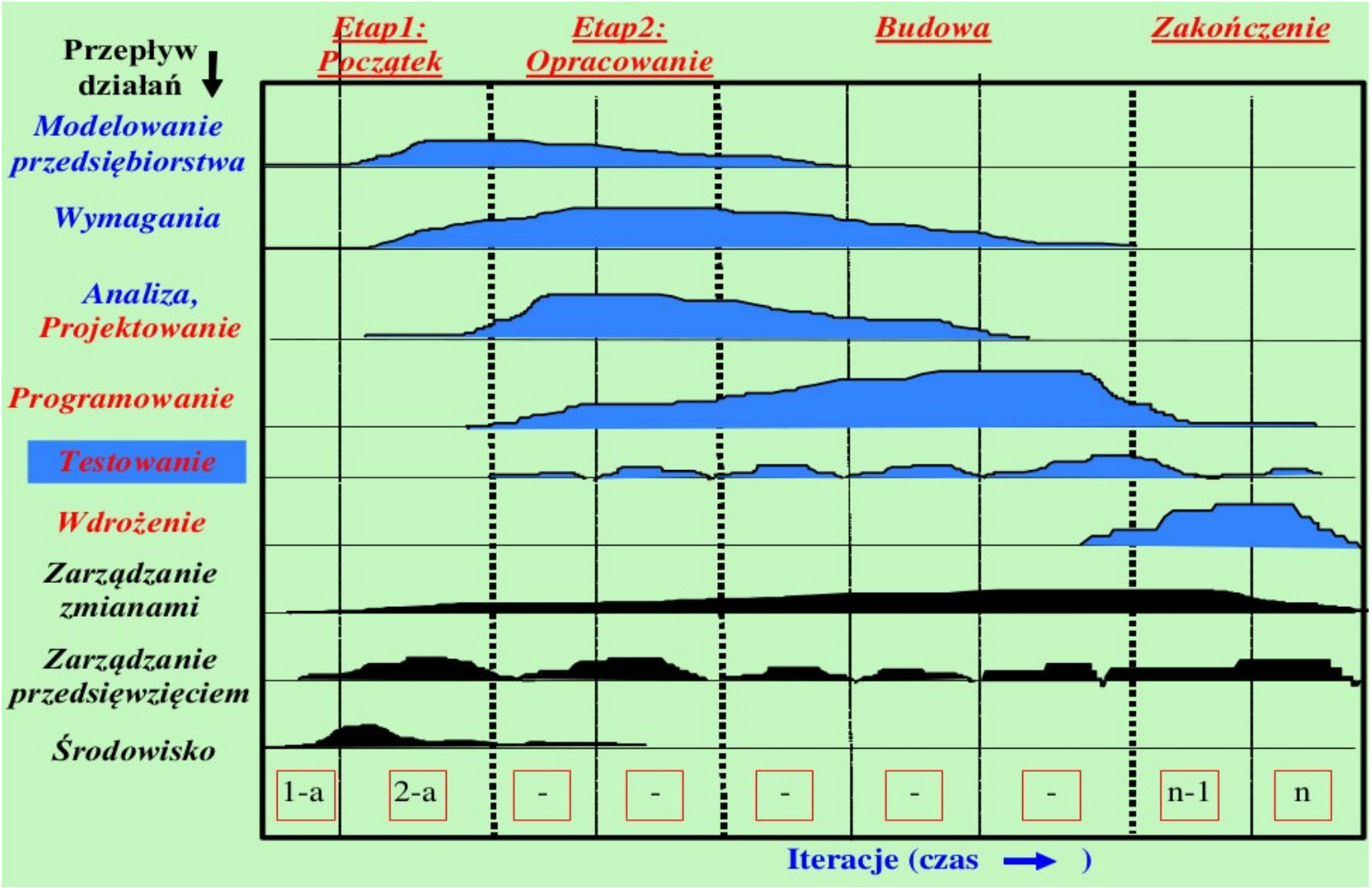
  specyfikacja wymagań → ogólny projekt → kolejne iteracje:
  (wybór komponentu → szczegółowy projekt komponentu
  → implementacja komponentu → testowanie → wdrożenie)
  → konserwacja całego SI

- **Zalety modelu:**
  - stopniowe testowanie, walidacja i wdrażanie (komponent po komponencie) – a nie dopiero na końcu procesu.
- **Wady modelu:**
  - dodatkowy koszt niezależnej realizacji komponentów SI.

Model przyrostowy

- Co i kiedy?

Źródło: materiały wykładowe dr inż. Zofii Kruczkiewicz



Model spiralny

- Ewolucyjny model z cykliczną kolejnością realizacji etapów:

  specyfikacja wymagań → projektowanie → implementacja → testowanie → wdrożenie

- **Zalety modelu:**
  - w kolejnym cyklu można wrócić do dowolnego etapu i „naprawić” co trzeba,
  - kolejne etapy projektu uwzględniają rozwiązania alternatywne i ryzyko realizacji.

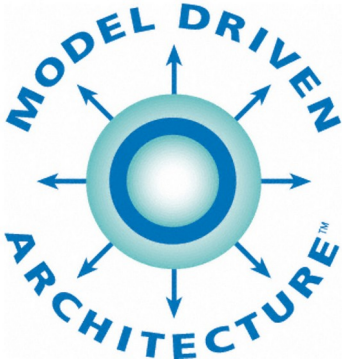
Wybrane inne modele

- **Prototypowanie:** wstępne wymagania → prototyp SI → doprecyzowane wymagania → właściwy SI (dla nowatorskich rozwiązań)
- **Programowanie odkrywcze:** wstępne wymagania → wdrożenie prostego SI → odkrycie nowych wymagań przez klienta → bardziej złożony SI (itd.)

4

Architektura Sterowana Modelem

Architektura Sterowana Modelem
/Model Driven Architecture, MDA/

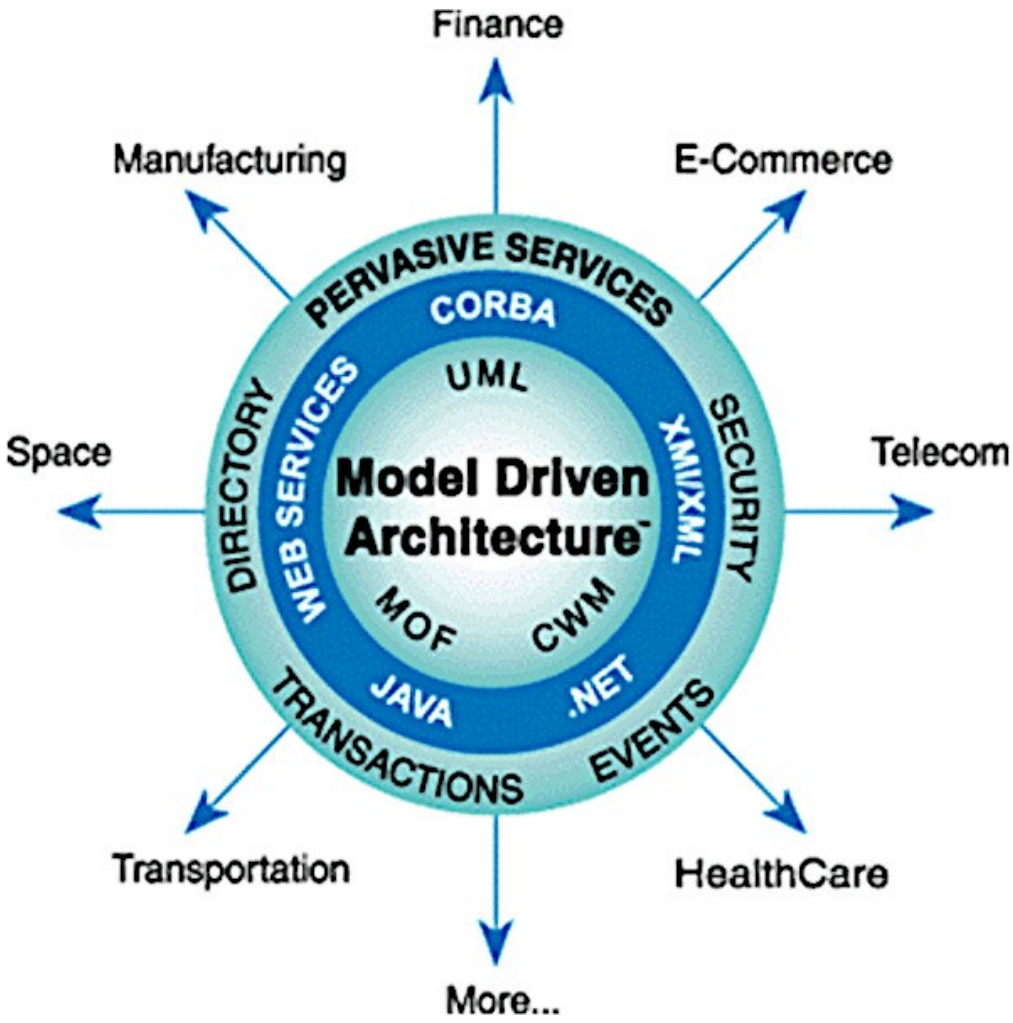


na podst. MDA - The Architecture Of Choice For A Changing World

- Metodyczne podejście do inżynierii oprogramowania (projektowania, rozwijania i implementowania oprogramowania):
  - oparte na budowie modeli (abstrakcji systemu) i ich transformacji,
  - opracowane przez *OMG Standards Development Organization.*
- Wytyczne strukturyzacji specyfikacji oprogramowania (w tym systemu zintegrowanego)
  - przy pomocy modeli niezależnych od platformy programistycznej (w tym Web Services, .NET, CORBA R, J2EE)
- Graficzne modelowanie funkcjonalności i biznesowego zachowania:
  - głównie w języku UML,
  - niezależnie od wybranej technologii ich implementacji.

Architektura Sterowana Modelem
/Model Driven Architecture, MDA/

- Izolacja rdzenia oprogramowania od techniki jego implementacji i cyklu zmian:
  - logika biznesowa pozostaje zgodna z biznesowymi wymaganiami,
  - a technologia oprogramowania może się rozwijać i zmieniać.
- Interoperacyjność w ramach i poza platformą programistyczną.
- **„Wszystko jest modelem”**
  - zamiast klasycznego podejścia do programowania, gdzie *„wszystko jest obiektem”.*



5

Poziomy abstrakcji modelu

Poziomy abstrakcji modelu

Od koncepcji do wdrożenia

1. Koncepcja

2. Model konceptualny

3. Model implementacyjny

4. Implementacja

5. Wdrożenie

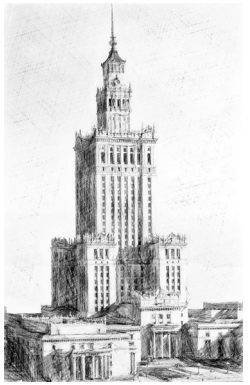
MDA

Kolejne poziomy abstrakcji modelu

- **Od koncepcji do szczegółowej implementacji systemu:**
  - **CIM** /Computation-Independent Model/ – model niezależny od środków informatycznych,
  - **PIM** /Platform-Independent Model/ – model niezależny od platformy programistycznej i systemowej,
  - **PSM** /Platform-Specific Model/ – model specyficzny dla platformy programistycznej i systemowej.
- Na podstawie modelu PSM powstaje kod programu.

Model CIM

- **CIM** /Computation-Independent Model/.
  - Model niezależny od środków informatycznych.
  - Koncepcja funkcjonalności systemu.
  - Nie uwzględnia, jakimi środkami informatycznymi (techniki implementacji, języka programowania itp.) wykonać system.
- CIM modeluje m.in.:
  - język dziedzinowy,
  - środowisko i użytkowników systemu,
  - wymagania funkcjonalne i niefunkcjonalne,
  - procesy biznesowe.



- Na podstawie wymagań i procesów biznesowych powstają przypadki użycia.

Model PIM

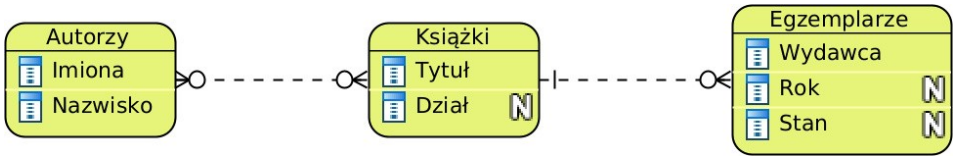
- **PIM** /Platform-Independent Model/.
  - Model niezależny od platformy programistycznej i systemowej.
  - Konceptualny model systemu – logika jego działania.
  - Uwzględnia środki informatyczne do wykonania systemu.
  - Nie uwzględnia, w jakich technologiach zaimplementować system.
  - Abstrakcyjna specyfikacja systemu.
- PIM modeluje m.in.:
  - podział systemu na komponenty i relacje między nimi;
  - budowę, zachowanie i współpracę komponentów;
  - artefakty używane i tworzone przez system.

Model PSM

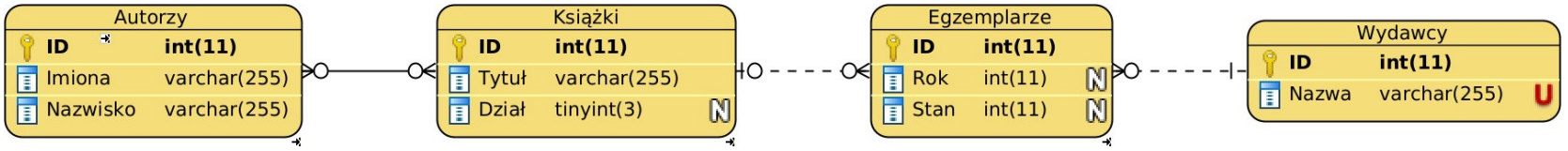
- **PSM** /Platform-Specific Model/.
  - Model specyficzny dla platformy programistycznej i systemowej.
  - Implementacyjny model systemu – sposób jego implementacji.
  - Uwzględnia środki informatyczne do wykonania systemu.
  - Uwzględnia, w jakich technologiach zaimplementować system.
  - Konkretna specyfikacja systemu.
- PSM modeluje m.in.:
  - wybór języka programowania;
  - wybór systemu i środowiska uruchamiania;
  - uzupełnienie modelu systemu o sprawy techniczne;
  - zapewnienie wydajności, niezawodności i bezpieczeństwa systemu.
- Na podstawie modelu PSM powstaje kod programu.

MDA w modelowaniu bazy danych

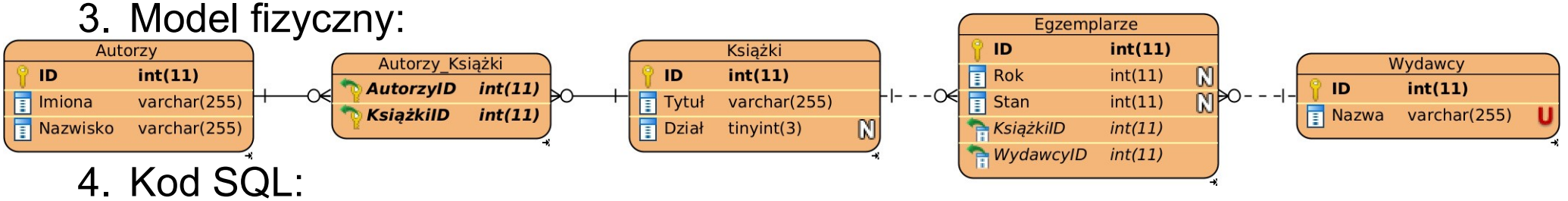
1. Model konceptualny:



2. Model logiczny:



3. Model fizyczny:



4. Kod SQL:

```
CREATE TABLE Autorzy (
        ID int(11) NOT NULL AUTO_INCREMENT,
        Imiona varchar(255) NOT NULL,
        Nazwisko varchar(255) NOT NULL,
        PRIMARY KEY (ID));

CREATE TABLE Książki (
        ID int(11) NOT NULL AUTO_INCREMENT,
        Tytuł varchar(255) NOT NULL,
        Dział tinyint(3),
        PRIMARY KEY (ID));

CREATE TABLE Autorzy_Książki (
        AutorzyID int(11) NOT NULL,
        KsiążkiID int(11) NOT NULL,
        PRIMARY KEY (AutorzyID, KsiążkiID));
```

```
ALTER TABLE Autorzy_Książki ADD CONSTRAINT
        FKAutorzy_Ks998910 FOREIGN KEY (AutorzyID)
        REFERENCES Autorzy (ID);

ALTER TABLE Autorzy_Książki ADD CONSTRAINT
        FKAutorzy_Ks72080 FOREIGN KEY (KsiążkiID)
        REFERENCES Książki (ID);
```

Poziomy abstrakcji modelu

W MDA wykorzystuje się

- **Meta-meta-model** (MOF) – definicja, jak modelować i jak zarządzać modelami.
- **Model dziedziny** – jednoznaczne zdefiniowanie pojęć dziedzinowych (ontologicznych), np. w języku OWL (*Web Ontology Language*)
  - etap CIM.
- **Metamodel** – elementy języka UML i powiązanych języków.
- **Model konceptualny** – abstrakcyjna definicja struktury i zachowania systemu przy pomocy diagramów UML (i nie tylko)
  - etap PIM.
- **Wzorce projektowe** – jednoznaczne, powtarzalne elementy modelujące strukturę i zachowanie.
- **Model implementacyjny** – konkretna definicja struktury i zachowania systemu przy pomocy diagramów UML (i nie tylko)
  - etap PSM.
- **Model platformy** – sposób odwzorowania PSM na kod programu.

6

Etapy MDA

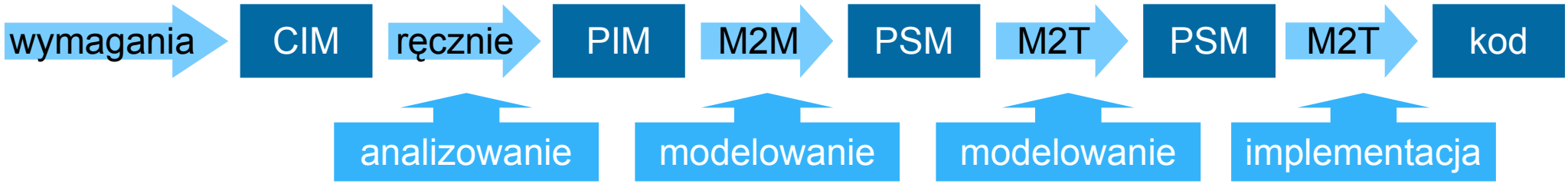
Etapy MDA

Kolejność etapów MDA w projektowaniu systemu

1. **Dziedzina problemu** (CIM).
   - Jak najwięcej pojęć dziedzinowych.
2. **Ogólna specyfikacja** systemu i budowa jego konceptualnego modelu (PIM).
   - Głównie w języku UML.
3. **Model platformy.**
   - W językach UML i OCL.
   - W repozytoriach MOF.
   - Każda platforma ma swój profil UML.
4. Odwzorowanie abstrakcji PIM na **konkretne technologie** PSM.
5. **Implementacyjny model** systemu (pół-/automatyczna transformacja PIM do PSM).
   - Transformacja PIM do PSM może być kilku-etapowa.
6. **Implementacja systemu** – pół-/automatyczna generacja kodu (zwykle prototypu).

Kolejność etapów MDA w projektowaniu systemu

- Łańcuch transformacji modelu w model (M2M) i modelu w tekst (M2T):
  - **CIM** → **PIM** → **PSM** → **tekst** (np. kod programu)



- Jeśli transformacje modelowe są poprawne, to:
  - mając „kompletną” i „poprawną” formalną specyfikację systemu,
  - dostaniemy „kompletny” i „poprawny” kod.
- Co znaczy „kompletny” i „poprawny”?

Temat następnej prezentacji

Język UML – diagramy zachowań