25. MDA – modele, idea, transformacje

**MDA** (Model Driven Architecture) jest specyfikacją projektowania i rozwoju projektów informatycznych, opracowaną przez OMG(Object Management Group). Podejście MDA zakłada następujące etapy budowy systemu informatycznego:

* Specyfikacja systemu w sposób niezależny od platformy implementacyjnej
* Wybranie najbardziej odpowiedniej platformy dla rozwijanego systemu
* Transformacja (odwzorowanie) systemu do wybranej platformy implementacyjnej

MDA to architektura, a nie metodyka – teoretycznie może być łączona z dowolną metodyką (ciężką i lekką)

Charakterystyka MDA:

* Kluczowe pojęcia: abstrakcja i model
* Stosowanie modeli na każdym etapie wytwarzania oprogramowania
* Kod systemu generowany automatycznie z modelu. W konsekwencji: wzrost abstrakcji tworzenia systemów względem tradycyjnych języków programowania (język modelowania – nowym językiem programowania)

Powiązane standardy:

* UML
* MOF (MetaObject Facility)
* XMI (XML Metadata Interchange)
* OCL
* QVT (Query View Transformation)

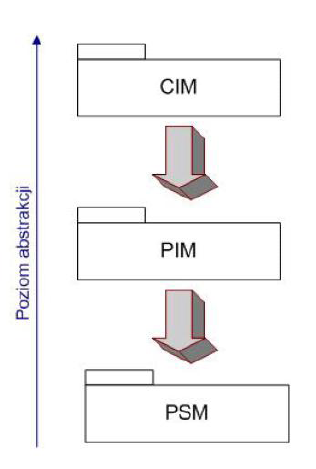
**Model** oprogramowania jest to abstrakcja systemu lub jego części. W zależności od typu modelu może on przedstawiać prosty lub bardziej szczegółowy widok systemu. Utworzone modele zapewniają efektywną komunikację między projektantami, zwłaszcza, gdy modelowany system jest bardzo złożony i wymaga zaangażowania wielu osób (zespołów). Poza tym modele tworzy się i można je zrozumieć o wiele szybciej niż np. tysiące linii kodu.

W skład MDA wchodzą następujące modele:

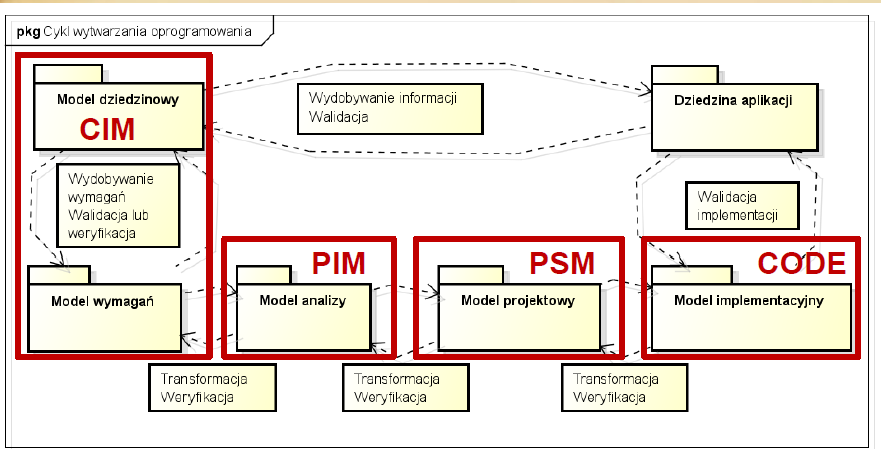
* **CIM** (computation independent model) – model niezależny obliczeniowo, odpowiednik modelu biznesowego oraz modelu przypadków użycia. model (zestaw modeli), który stanowi specyfikację wymagań systemu. Skupia się na środowisku, w którym system ma działać. Szczegóły struktury i przetwarzania danych przez system są ukryte albo jeszcze nieokreślone. Model ten jest niezależny od implementacji systemu, czasem nazywany modelem domeny albo modelem biznesowym. Na tym poziomie definiuje się wymagania funkcjonalne i niefunkcjonalne, reguły biznesowe itp.
* **PIM** (platform independent model) – model niezależny od platformy implementacyjnej, odpowiednik modelu analizy. model (zestaw modeli) pokazujący tę część systemu, która jest całkowicie niezależna od platformy sprzętowej i programowej, pod kontrolą której system będzie działał. Skupia się na działaniu systemu, ukrywając szczegóły konieczne dla danej platformy. Na tym poziomie tworzy się np. w UML model danych, architekturę systemu, diagramy aktywności itp.
* **PSM** (platform specific model) – model zależny od platformy, odpowiednik modelu projektowego. model (zestaw modeli), który powstaje po uzupełnieniu modelu PIM o informacje specyficzne dla platformy sprzętowej i programowej wykorzystywanej do działania systemu.

**+ Implementation Model = Code** – kod źródłowy generowany automatycznie z modelu PSM.

Powyższe modele reprezentują system z trzech różnych perspektyw i na różnych poziomach abstrakcji (od najbardziej ogólnego do najbardziej szczegółowego).



**MDA a klasyczny cykl wytwarzania oprogramowania:**



**Transformacja** w MDA polega na automatycznej generacji modelu wyjściowego z modelu wejściowego wykonanej w oparciu o definicję transformacji. **Definicja transformacji** jest to zbiór **reguł transformacji**, czyli opisów przekształceń jednego lub więcej elementów języka modelu wejściowego w jeden lub więcej elementów modelu wyjściowego.

Sposoby transformacji PIM do PSM:

* Odwzorowanie elementów metamodelu PIM do elementów metamodelu PSM
* Wzbogacenie modelu PIM specjalnymi znacznikami (np. stereotypami) i transformacja tak oznakowanego modelu PIM do PSM

**Klasyfikacja transformacji:**

* Transformacje wertykalne (modele wejściowy i wyjściowy na innym poziomie abstrakcji)
  + Inżynieria do przodu
  + Inżynieria odwrotna
* Transformacje horyzontalne (modele wejściowy i wyjściowy na tym samym poziomie abstrakcji):
  + Migracja (zmiana języka, bez zmiany poziomu abstrakcji)
  + Restrukturyzacja (refaktoryzacja)

Transformacje horyzontalne zwykle zachodzą z dwóch powodów:

* + - Poprawa jakości atrybutów modelu (refaktoryzacja)
    - Wsparcie analizy modelu

**Metody transformacji pomiędzy modelami:**

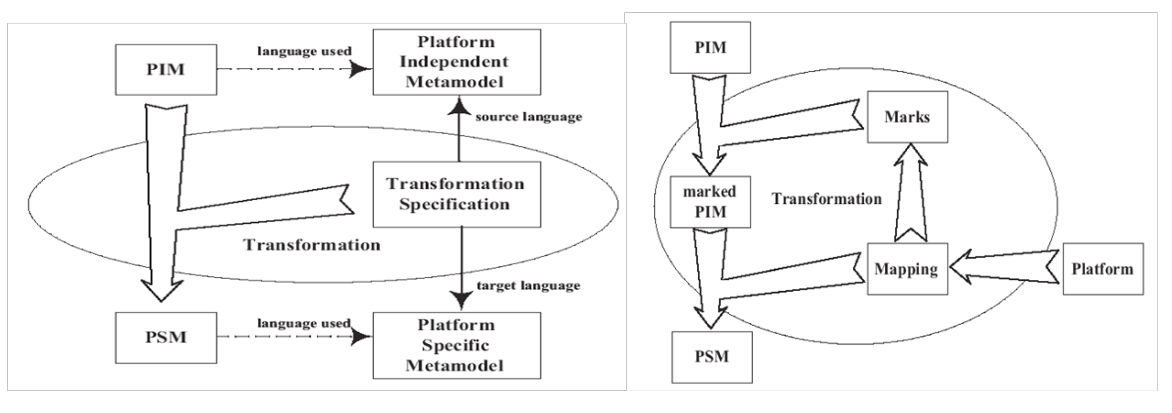
* Manipulacja bezpośrednia
* Metody relacyjne
* Transformacja grafów
* Transformacje wykorzystujące XSLT (język przekształceń - Zawiera elementy służące do definiowania reguł opisujących sposób przekształcania jednego dokumentu XML na inny dokument; Transformacja XSLT to automatyczna konwersja dokumentu XML do formatu HTML, WML, itp. lub innego dokumentu XML)
* Metody hybrydowe

**Metody transformacji pomiędzy modelem i kodem:**

* Metody wykorzystujące wzorzec wizytator
* Metody oparte o szablony

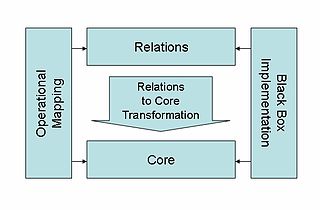
**Sposoby transformacji (odwzorowania) PIM do PSM:**

* Odwzorowanie elementów metamodelu PIM do elementów metamodelu PSM
* Wzbogacenie modelu PIM specjalnymi znacznikami (np. stereotypami) i transformacja tak oznakowanego modelu PIM do PSM



Aby umożliwić automatyczną generację kodu Object Management Group stworzył standard o nazwie MOF Model to Text Transformation Language, dzięki któremu możliwa jest transformacja PSM w takie artefakty jak: wykonywalny kod, raporty, dokumenty, specyfikacja wdrożenia. Standard ten opisuje reguły transformacji modelu oprogramowania w postać tekstową. Oparty jest on na szablonach tekstowych parametryzowanych częściami modelu.

**Standard QVT (Query-View-Transformation)**

* Jest standardem dla modelu transformacji, a ściślej mówiąc QVT jest językiem pozwalającym zaprojektować automatyczne transformacje między modelami UML. Transformacje raz stworzone mogą zostać np. zapisane w języku QVT i wykorzystane później w innych projektach.
* Standardowy zbiór języków specjalnie przeznaczonych do definiowania transformacji, zdefiniowany przez OMG.
* Definiuje on 3 języki transformacji modelowej. Wszystkie z nich operują na meta modelach zdefiniowanych w MOF 2.0 (MOF = Meta-Object Facility - określa sposób, standardy modelowania oraz zarządzania modelami). Transformacja wykonana w dowolnym z tych 3 języków może być uważana za model zgodny z metamodelami zdefiniowanymi w standardzie MOF. Standard QVT integruje OCL 2.0 i rozszerza go o elementy języka imperatywnego (program jako sekwencja instrukcji zmieniających stan programu).
* **QVT-Operational** – język imperatywny zaprojektowany do pisania transformacji jednokierunkowaych.
* **QVT-Relations** – język deklaratywny (programista opisuje warunki, jakie musi spełniać końcowe rozwiązanie -*co chcemy osiągnąć*- a nie szczegółową sekwencję kroków, które do niego prowadzą -*jak to zrobić*) zaprojektowany aby zezwolić na pisanie zarówno jednokierunkowych, jak i dwukierunkowych transformacji. Transformacja w tym języku uwzględnia spójność zbioru modeli. Spójność ta może być sprawdzana poprzez wywołanie transformacji w trybie (check-only) – zwraca ona true gdy zbiór modeli jest spójny i false w.p.p. Ta sama transformacja może być też użyta do modyfikacji jednego z modeli – tak, że zbiór modeli będzie spójny. Jezyk QVT-Relations – składna tesktowa i graficzna.
* **QVT-Core** – prosty język deklaratywny, wynik transformacji z języka QVT-Relations.
* Mechanizm QVT-BlackBox służy do wywoływania transformacji na modelach w innych językach (np. XSLT, XQuery)
* Języki QVT nie zezwalają na transformację z lub do modeli tekstowych. Transformacje model-to-text są standaryzowane osobno przez OMG (MOFM2T)

**Przykładowe narzędzia wspierające MDA:**

* Together (model-model),
* ArcStyler (model-kod),
* Visual Paradigm (model-kod, model-model)
* Eclipse:
  + Eclipse Modeling Framework (EMF)
  + ATL Development Tools (model-model) – język Atlas
  + MOFScript (model-kod)
  + Acceleo (model-kod)
  + AndroMDA (model-kod)

**Co daje stosowanie MDA?**

* Wieloużycie projektów na poziomie analitycznym
  + Redukcja kosztów przenoszenia oprogramowania
  + Przenośność na platformy, które jeszcze nie istnieją

**MDD**

* Wytwarzanie oprogramowania sterowane modelami (MDD – Model Driven Development) – uogólnienie podejścia MDA
* Elementem centralnym wytwarzania jest model
* Model może być utworzony w dowolnym języku modelowania, np. w dedykowanym do danego celu języku dziedzinowym
* Model jest przekształcany zwykle bezpośrednio do kodu (w podejściu MDD nie buduje się kaskady modeli)