

# Основные концепции модельно-центрированной разработки (MDA, MOF, XMI)

# Модельно-центрированная разработка

Model-driven development = стиль разработки, при котором главный артефакт - модель

## **Modelware**

## **Model-driven architecture**

Модель - единый сложно устроенный артефакт

Моделей много

Модель уточняется и детализируется

Новые модели получаются из существующих путем трансформации

Цель моделирования - работающая (программная) система

- ***Модель*** описание или спецификация системы и ее окружения, созданная для определенных целей. Часто является комбинацией текстовой и графической информации. Текст может быть описан специализированным или естественным языком.

# MDA

## Архитектура, управляемая моделью (Model Driven Architecture, MDA)

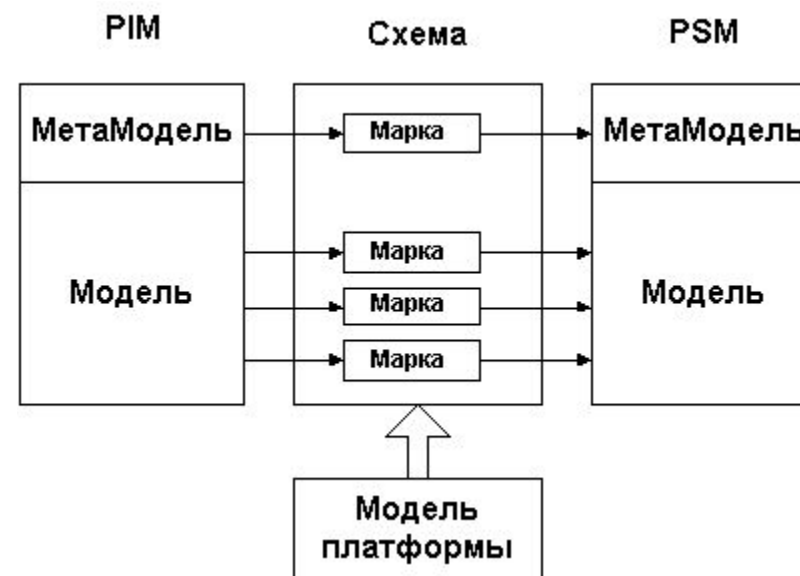
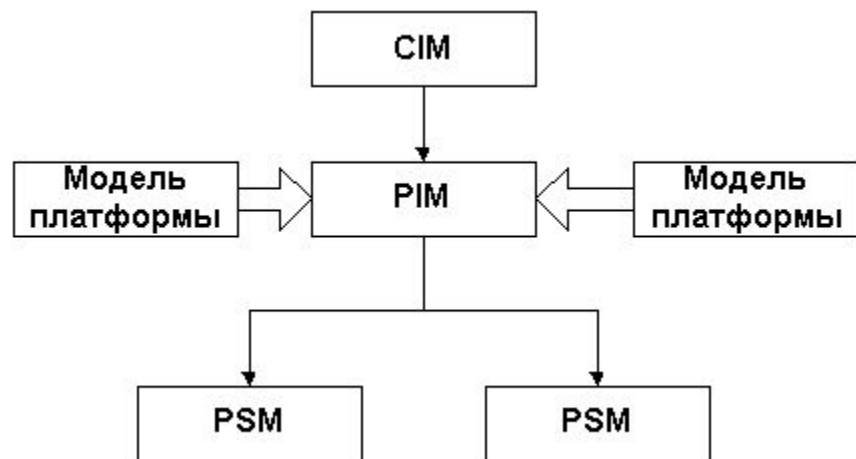
- создаваемая консорциумом OMG\* разновидность концепции «Разработка, управляемая моделями»: модельно-ориентированного подхода к разработке программного обеспечения.
- Его суть состоит в построении абстрактной метамодели управления и обмена метаданными (моделями) и задании способов ее трансформации в поддерживаемые технологии программирования (Java, CORBA, XML и др.).

# Object Management Group

- **OMG** (читается как [о-эм-джи]) — консорциум (рабочая группа), занимающийся разработкой и продвижением объектно-ориентированных технологий и стандартов. Это некоммерческое объединение, разрабатывающее стандарты для создания интероперабельных, то есть платформо-независимых, приложений на уровне предприятия. С консорциумом сотрудничает около 800 организаций — крупнейших производителей программного обеспечения.

# Основные идеи MDA

- Для конструирования программного приложения должна быть построена подробная, формально точная модель, из которой потом может быть автоматически генерирован исполняемый программный код приложения.



# Основные шаги разработки:

- Сначала разрабатывается модель предметной области проектируемого приложения, полностью независимая от имплементирующей технологии. Она называется Platform Independent Model (PIM).
- Затем PIM автоматически трансформируется специальным инструментом в платформу-зависимую модель (Platform Specific Model, PSM).
- PSM переводится в исходный код на соответствующем языке программирования.

В реальных современных проектах часть бизнес-логики приходится реализовать вручную. Но поскольку этот код отделен от генерированного системой, большой проблемы это не представляет.

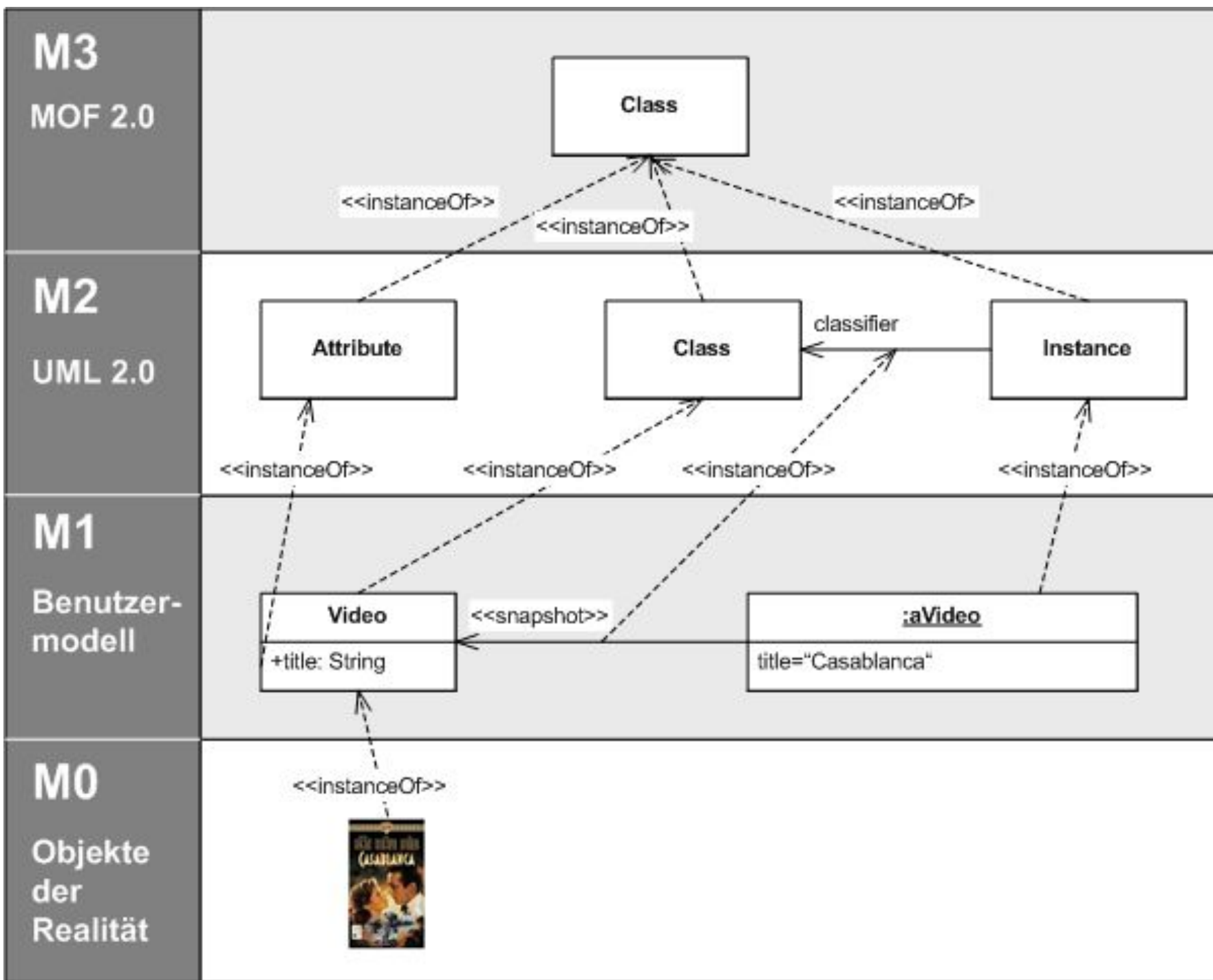
- **Платформа** набор подсистем и технологий, которые представляют собой единый набор функциональности, используемой любым приложением без уточнения деталей реализации.
- **Платформенно-независимая модель (PIM)** модель, скрывающая детали реализации системы, зависящие от платформы, и содержащая элементы, не изменяющиеся при взаимодействии системы с любой платформой.
- **Платформенно-зависимая модель (PSM)** модель системы с учетом деталей реализации и процессов, зависящих от конкретной платформы.
- **Платформенная независимость** качество модели, обозначающее ее независимость от свойств любой платформы.
- **Вычислительно-независимая модель (Computation Independent Model, CIM)** описывает общие требования к системе, словарь



# MOF

## Мета-объектное средство (Meta-Object Facility)

- Это стандарт для разработки, управляемой моделями, разработанный OMG.
- MOF возникло из UML. OMG нуждался в архитектуре мета-моделирования для определения UML. MOF реализовано как четырехслойная архитектура. Ядром всего проекта является мета-мета модель M3 на верхнем уровне. Она определяет язык, используемый MOF для создания метамodelей, называемых *M2-моделями*. Наиболее ярким примером модели 2-го уровня MOF является метамодель UML: модель, которая описывает сам язык UML. Эти M2-модели описывают элементы M1 слоя: M1-модели. Это могут быть, например, модели, написанные на UML. Последний слой — M0-слой или слой данных. Он используется для описания объектов реального мира.



# MOF - это *закрытая* архитектура мета-моделирования;

Определены два варианта MOF:

- **EMOF (Essential MOF)** — подмножество MOF 2.0, которое служит для того, чтобы создавать простые метамодели простыми средствами - без необходимости понимать MOF в полном варианте. EMOF в очень большой степени совместим с распространённой метамоделью ECore, определённой в Eclipse Modeling Framework.

- **CMOF (Complete MOF)** содержит полный объем языковых средств.

# XMI

стандарт для обмена метаданными с помощью языка XML

- **XMI** (*XML Metadata Interchange*):

Может использоваться для любых метаданных, если их метамодель может быть выражена с помощью MOF (Meta-Object Facility). Наиболее часто XMI применяется как формат обмена UML-моделями.

Информатик не мог раньше импортировать UML-модель из одного инструмента UML-моделирования в другой — из-за различий в определении синтаксиса и семантики элементов языка.

# Синтаксис XMI

- Спецификация XMI довольно сложна (в версии 1.2 — более 400 страниц).

Метамоделю языка UML, изложенная на XMI, также пугающе велика. Чтобы дать общее представление о синтаксисе, воспользуемся тем фактом, что на XMI определяется не только метамодель языка UML, но и конкретные UML-модели.

Обратите внимание, что тэги "UML:Class", "UML:Attribute", "UML:Classifier" не принадлежат языку XMI, а были определены в метамодели языка UML, на которую ссылается данный XMI-файл.

# Пример XMI-файла: адрес

```
<?xml version="1.0"?> <XMI xmi.version="1.2" xmlns:UML="org.omg/UML/1.4">
```

```
<XMI.header>  
  <XMI.documentation>  
    <XMI.exporter>ananas.org stylesheet</XMI.exporter>  
  </XMI.documentation>  
  <XMI.metamodel xmi.name="UML" xmi.version="1.4"/>  
</XMI.header>  
<XMI.content>  
  <UML:Model xmi.id="M.1" name="address" visibility="public"  
    isSpecification="false" isRoot="false"  
    isLeaf="false" isAbstract="false">  
    <UML:Namespace.ownedElement>  
      <UML:Class xmi.id="C.1" name="address" visibility="public"  
        isSpecification="false" namespace="M.1" isRoot="true"  
        isLeaf="true" isAbstract="false" isActive="false">  
        <UML:Classifier.feature>  
          <UML:Attribute xmi.id="A.1" name="name" visibility="private"  
            isSpecification="false" ownerScope="instance"/>  
          <UML:Attribute xmi.id="A.2" name="street" visibility="private"  
            isSpecification="false" ownerScope="instance"/>  
          <UML:Attribute xmi.id="A.3" name="zip" visibility="private"  
            isSpecification="false" ownerScope="instance"/>  
          <UML:Attribute xmi.id="A.4" name="region" visibility="private"  
            isSpecification="false" ownerScope="instance"/>  
          <UML:Attribute xmi.id="A.5" name="city" visibility="private"  
            isSpecification="false" ownerScope="instance"/>  
          <UML:Attribute xmi.id="A.6" name="country" visibility="private"  
            isSpecification="false" ownerScope="instance"/>  
        </UML:Classifier.feature>  
      </UML:Class>  
    </UML:Namespace.ownedElement>  
  </UML:Model>  
</XMI.content>
```

```
</XMI>
```

# **XMI-шапка:** XMI определяет следующие теги и атрибуты:

XMI всегда является корневым элементом. Он должен иметь атрибут `xmi.version` (действующие версии 1.0, 1.1, 1.2 и 2.0).

- XMI.header — шапка. Наиболее важны ее дети XMI.documentation и XMI.metamodel.
  - XMI.documentation содержит информацию о конечном пользователе:
    - XMI.owner — владелец метамодели.
    - XMI.contact — его контактные данные.
    - XMI.longDescription — длинное описание владельца.
    - XMI.shortDescription — короткое описание.
    - XMI.exporter — экспортер.
    - XMI.exporterVersion — версия экспортера.
    - XMI.exporterID — идентификатор экспортера.
    - XMI.notice — комментарий.
  - XMI.metamodel — описание метамодели, к которой алгоритм XMI был применен.
- XMI.content — содержание модели.
  - `xmi.id` — уникальный идентификатор ссылки на метамодель.
  - `xmi.idref` — сама ссылка на метамодель.

Благодарю за  
внимание!