Model-Driven Engineering

Model-Driven Engineering

Douglas C. Schmidt,
IEEE Computer, Feb. 2006

Model-Driven Development

- MDD shifts the development focus from code to models.

- A major goal of MDE research is to produce technologies that shield software developers from the complexities of the underlying implementation platform.

Model-Driven Architecture

Transforming Software Development:
An MDA Road Map

Thomas O. Meservy, Kurt D. Fenstermacher,
IEEE Computer, Sep. 2005

Models

- Models are used to flexibly represent complex systems
  - Models can be viewed at many levels of abstraction
  - Complementary model views can be combined to give a more intelligible, accurate view of a system than a single model alone

- Development teams commonly employ models only in the early stages of modeling
  - Once construction begins, the teams leave the model behind

MDA

- 2001: OMG launches the MDA initiative
- Goal: shifting the focus of software development from writing code to modeling

- MDA is part of a larger trend to carefully layer additional levels of abstraction onto the underlying hardware
  - High-level languages have replaced assembly
  - Libraries and frameworks are replacing isolated code
  - Design patterns are replacing project-specific code

MDA

- Working code will always be the ultimate goal of software development
- MDA faces the substantial challenge of carrying models through to implementation
- MDA can help software development teams concentrate their efforts on modeling and generating much of the code needed

MDA in a Nutshell

- After first generating a set of requirements, developers build a system model that satisfies those requirements
- The initial model captures the requirements, but without committing to a specific technology platform.
- Using a set of rules, a MDD environment then transforms this platform-independent model (PIM) into a platform-specific model (PSM)

MDA in a Nutshell

- Because a PSM is still too abstract for compilation, an MDA environment needs a second set of transformations to map from the PSM to code.
- With MDA, the environment produces code from models that software developers write, rather than programmers writing the code directly
- As the system changes, developers modify the model and the environment synchronizes the code with the changed model.
- The model is no longer discarded at the outset of coding but instead is the focus of development effort.

MDA in Action

- MDA makes a sharp distinction between:
  - A model of the business (CIM)
  - The business model in a specific technology (PIM)
  - A model that uses platform-specific code (PSM)

- MDA development life cycle includes a five-step process:
  1. capture requirements in a CIM
  2. create a PIM
  3. transform the PIM to one or more PSMs
  4. transform the PSM to code
  5. deploy the system in a specific environment

Example

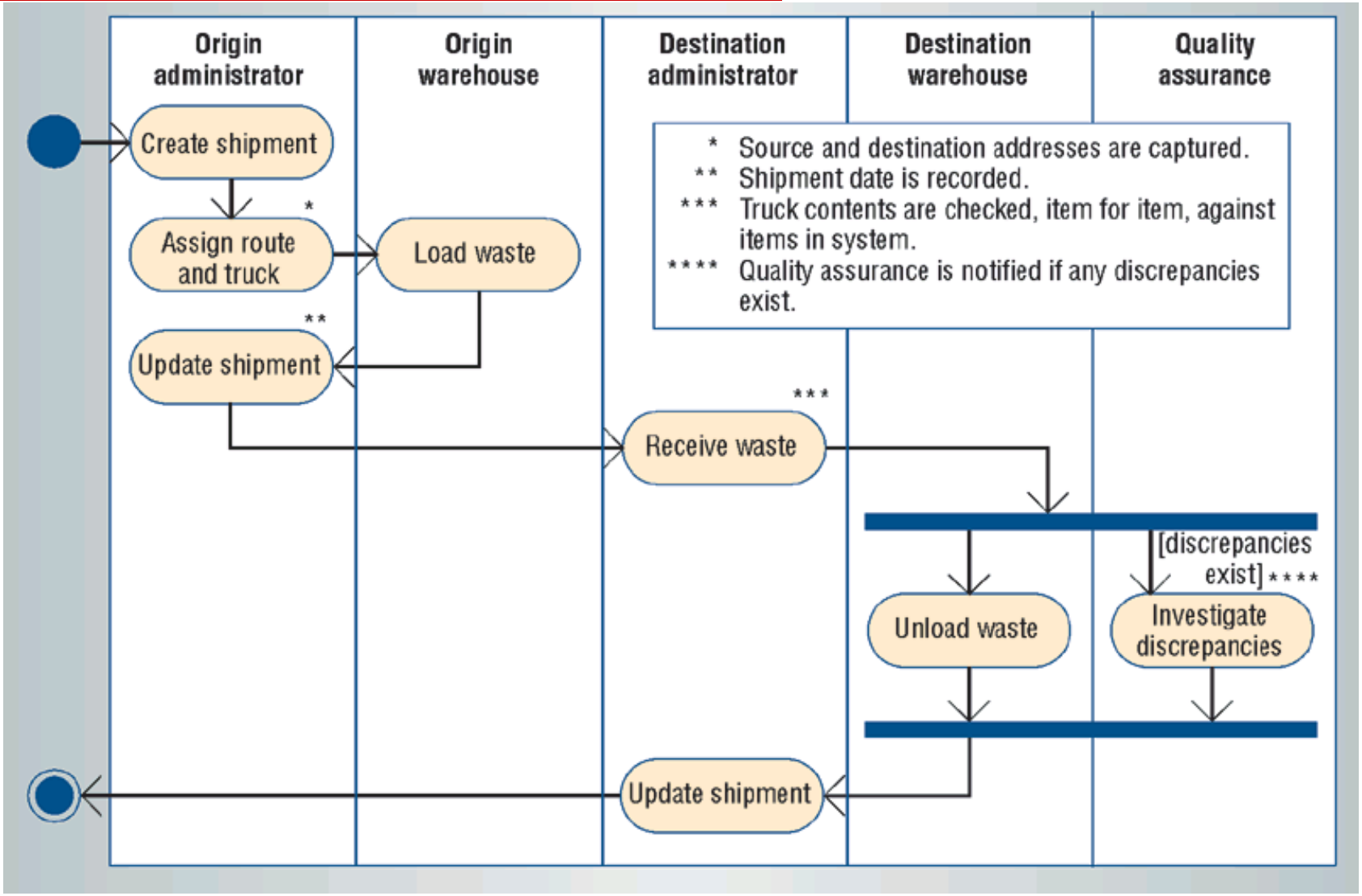
- Hazard County Waste is a shipping company that specializes in handling hazardous materials
- HCW recently won a substantial contract requiring it to provide the client a Web-based application to track chemical shipments
- HCW outsourced the project to Model-Driven Software Inc. (MDSI)

Capturing requirements

- The computation-independent model (CIM) captures the domain without reference to a particular system implementation or technology

- The CIM would remain the same even if the system were implemented mechanically, rather than in computer software.

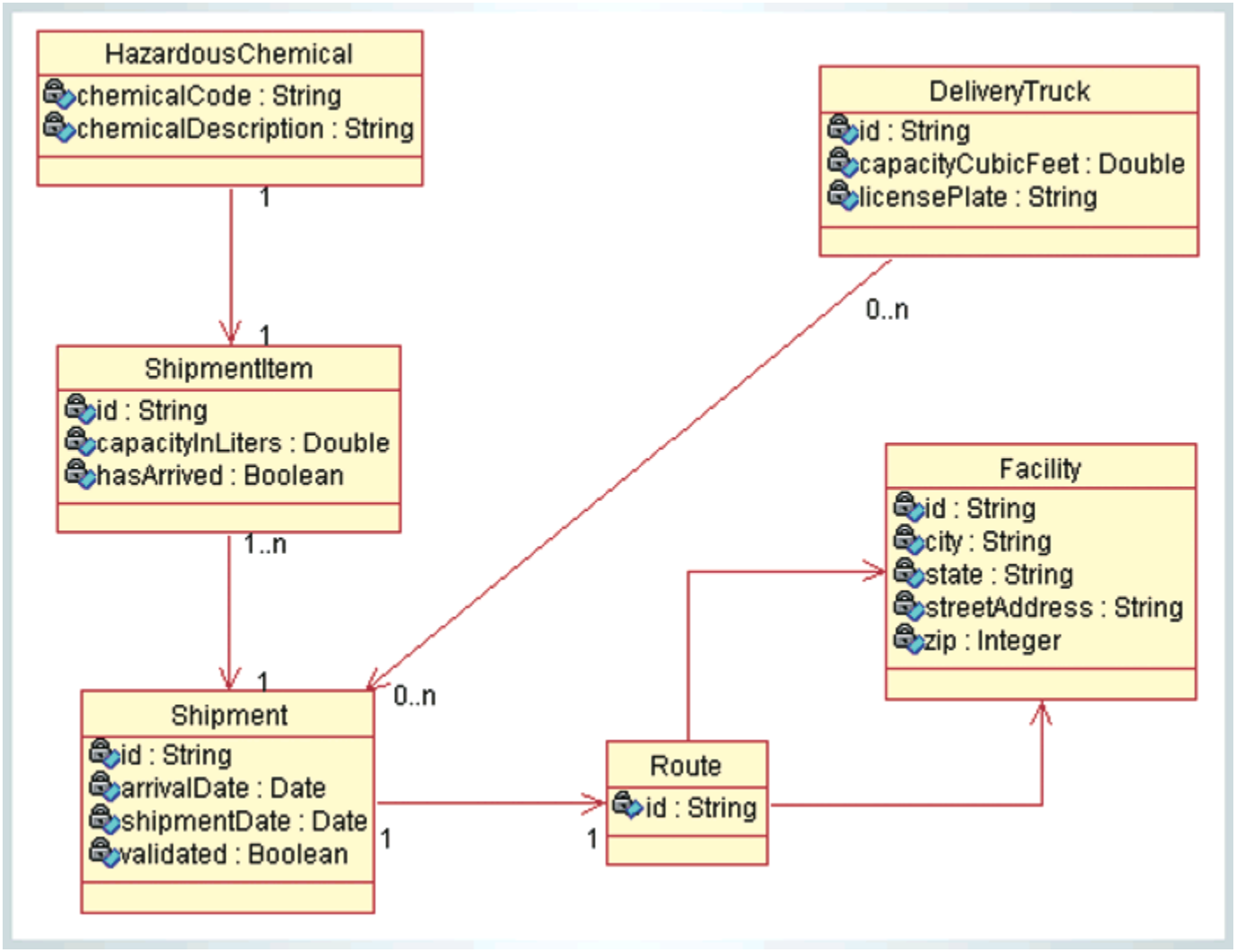
CIM


Origin administrator
Origin warehouse
Destination administrator
Destination warehouse
Quality assurance
Create shipment
Assign route and truck
Load waste
Update shipment
Receive waste
Unload waste
Investigate discrepancies
[discrepancies exist]
Update shipment
* Source and destination addresses are captured.
** Shipment date is recorded.
*** Truck contents are checked, item for item, against items in system.
**** Quality assurance is notified if any discrepancies exist.

Creating a PIM

- While not including any platform-specific details, the PIM should capture the domain's key concepts or semantics
- The goal is to generate a high-level dictionary for the project that captures abstract concepts such as "shipments consist of items."
- PIM demands a representation sufficiently general to capture the semantics of many different domains
- MDA uses UML as its core representation.

Creating a PIM



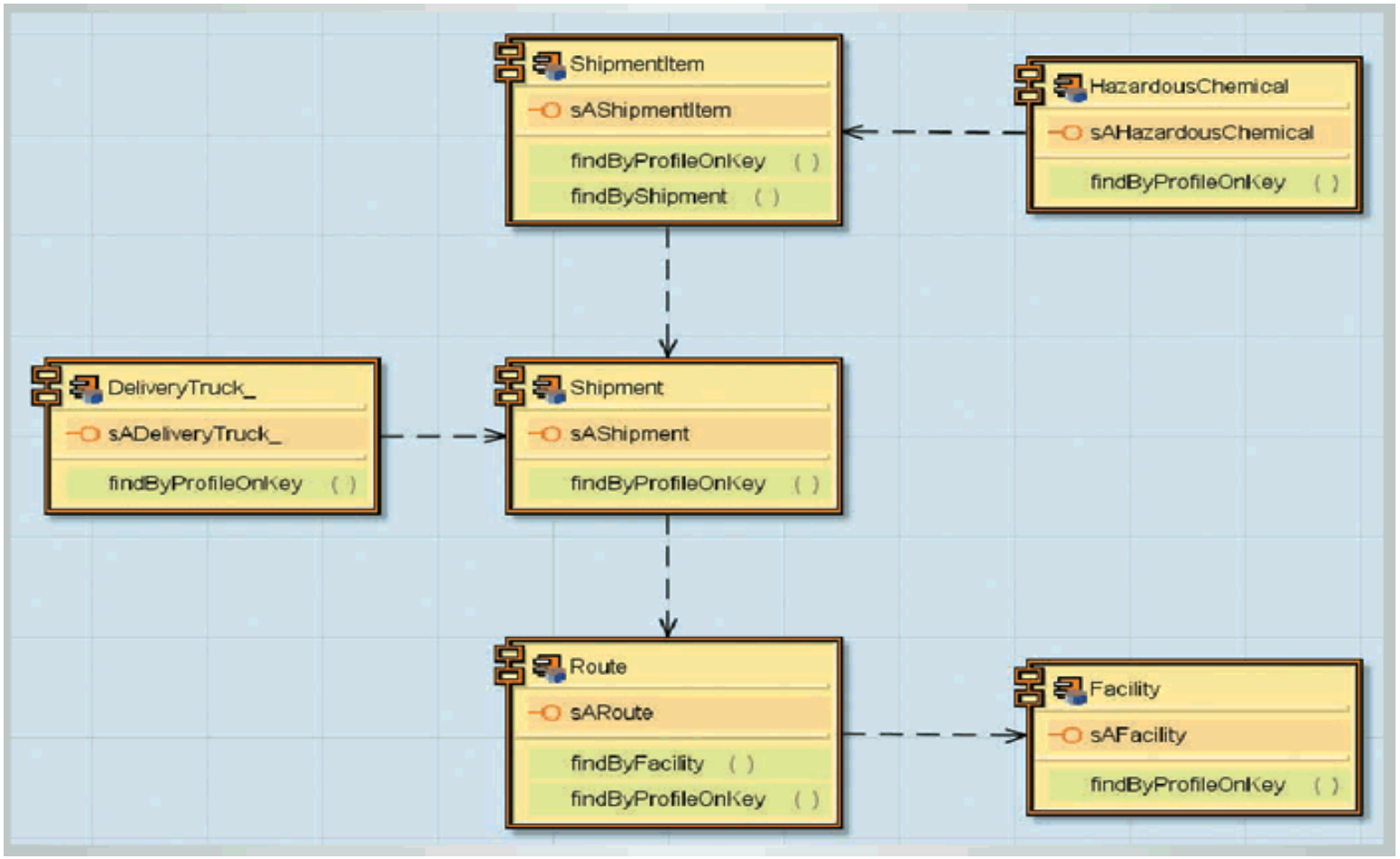
Creating a PIM

- MDSI opts in this case to build the PIM for HCW using a separate application, IBM's Rational Rose 2003
- This visual modeling software does not support transformation of the PIM to the PSM
- But it does enable exporting the PIM in XMI to another popular MDA tool, Compuware's OptimalJ

From PIM to PSM

- MDSI uses:
  - RDBMS to represent the persistence layer
  - EJB to represent the business layer
  - JSP to represent the presentation layer

PSM

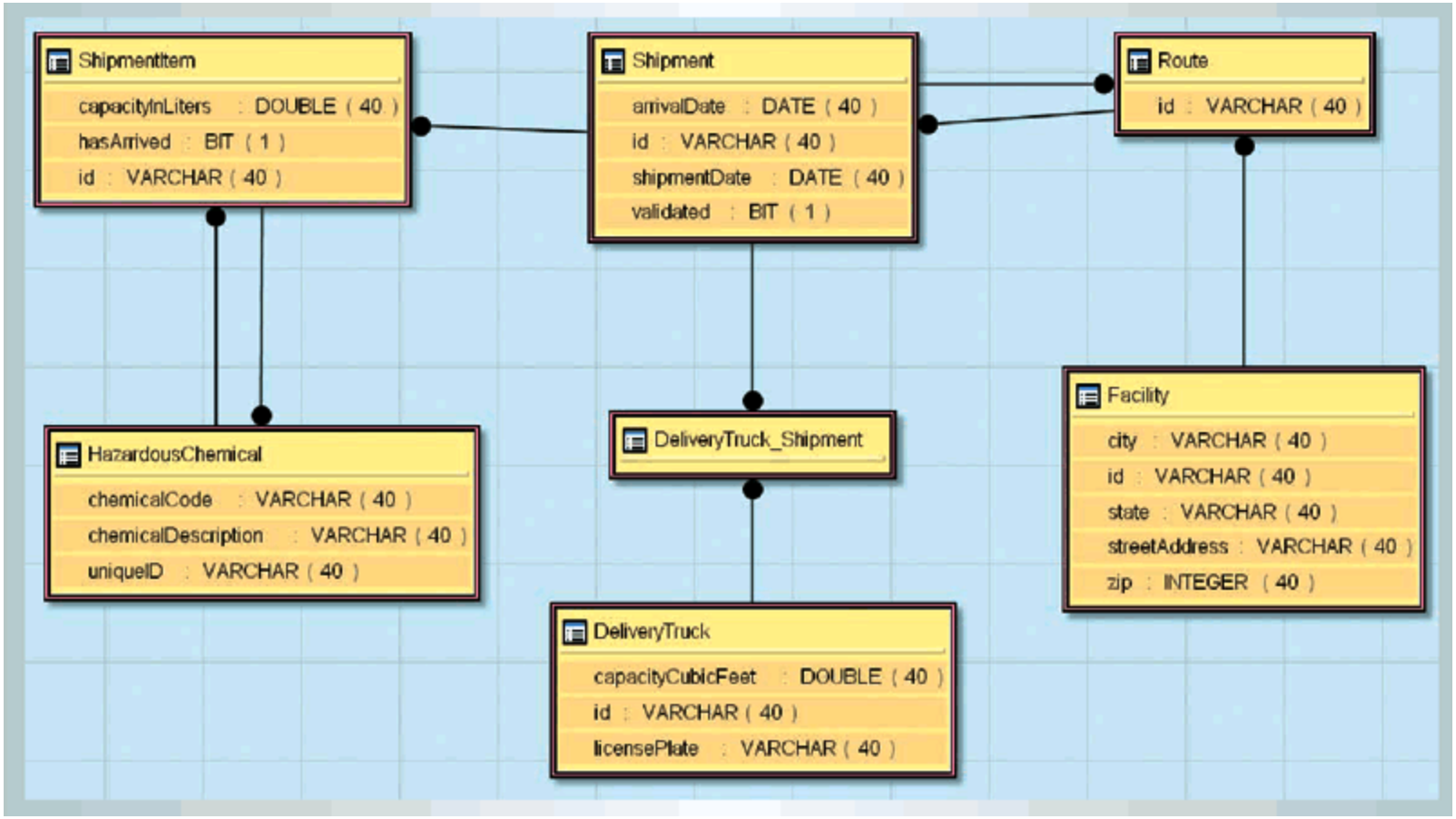

ShipmentItem
sAShipmentItem
findByProfileOnKey ( )
findByShipment ( )
HazardousChemical
sAHazardousChemical
findByProfileOnKey ( )
DeliveryTruck_
sADeliveryTruck_
findByProfileOnKey ( )
Shipment
sAShipment
findByProfileOnKey ( )
Route
sARoute
findByFacility ( )
findByProfileOnKey ( )
Facility
sAFacility
findByProfileOnKey ( )

From PIM to PSM

- Note that this model includes platform-specific details:
  - Two finder methods in the ShipmentItem class, one by key and another by shipment

- OptimalJ can transform a PIM into a platform-specific database model to support persistence.

PSM


ShipmentItem
capacityInLiters : DOUBLE (40)
hasArrived : BIT (1)
id : VARCHAR (40)
Shipment
arrivalDate : DATE (40)
id : VARCHAR (40)
shipmentDate : DATE (40)
validated : BIT (1)
Route
id : VARCHAR (40)
HazardousChemical
chemicalCode : VARCHAR (40)
chemicalDescription : VARCHAR (40)
uniqueID : VARCHAR (40)
DeliveryTruck_Shipment
Facility
city : VARCHAR (40)
id : VARCHAR (40)
state : VARCHAR (40)
streetAddress : VARCHAR (40)
zip : INTEGER (40)
DeliveryTruck
capacityCubicFeet : DOUBLE (40)
id : VARCHAR (40)
licensePlate : VARCHAR (40)

From PIM to PSM

- This type of model includes database-specific attributes and relationships
  - For example, the DeliveryTruck relation shows that it has a primary key named ID and that it participates in a many-to-many relationship with Shipment.
  - The model also lists specific data types for each relation.

From PSM to Code

- The next step involves transforming the PSM to code
- The PSM-to-code transformation is analogous to the PIM-to-PSM transformation

Evaluation

- Many critics, see MDA as the latest industry attempt at automating code generation and predict it will fail
  - Just as CASE did before it.

- However, CASE tools were generally proprietary, and so locked development organizations into specific vendors.
  - In contrast, MDA uses modeling and transformation languages that meet open standards

Evaluation

Evaluation

- The most common objection to MDA is that UML lacks sufficient precision to enable complete code generation.

- It is true that UML does not capture code-level semantics.

Evaluation

- Many software systems must address the myriad irregularities of real-world domains.
- If the PIM representation does not support sufficient detail to generate these irregularities at the code level, then the programmer must write code by hand
- Early CASE tools lacked the ability to easily insert such handwritten code into the automatically generated code and failed to support simple round-trip engineering of code

Evaluation

- Inserting handwritten code is especially important in MDA because the process is iterative, which means that MDA tools are continually regenerating code.

- Thus, MDA tools must distinguish between generated code and regions of handwritten code

- OptimalJ labels these as guarded and free blocks.
  - Guarded blocks contain generated code
  - Free blocks mark regions in the code that developers are expected to modify

Conclusions

- As with any new approach, MDA needs time to mature.
- We recognize that it is not a silver bullet, but it does offer important benefits for the software development process.
- At appropriate levels of abstraction, models can enhance customers' and developers' understanding of their problem domain.
- Models also can help translate that understanding into more reliable working code more quickly.