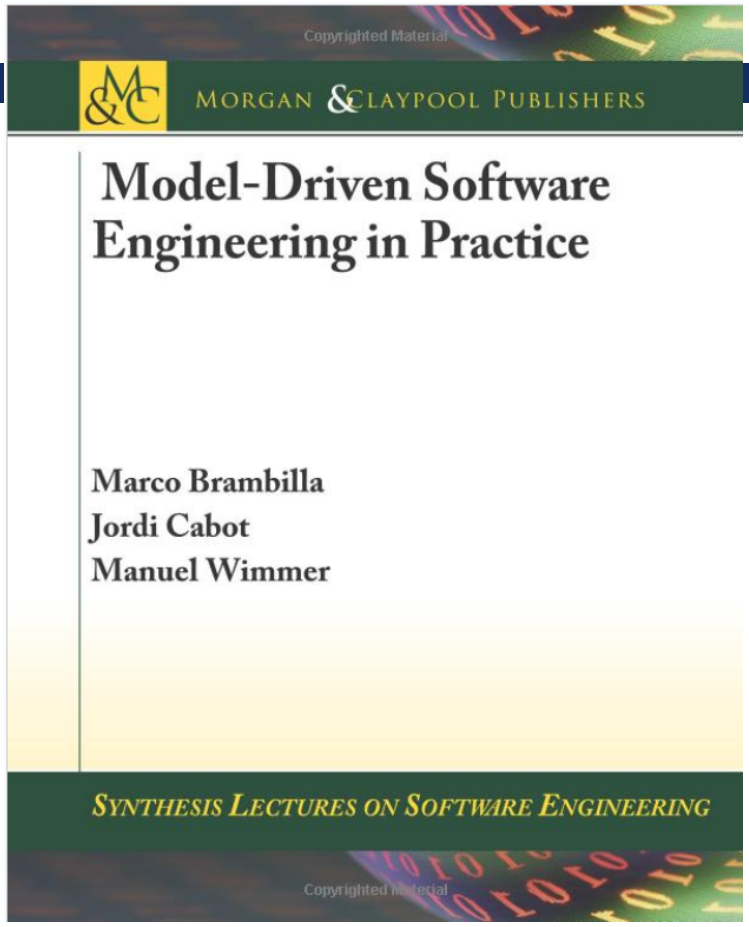
Model Driven Development

Shihong Huang
shihong@fau.edu
Department of Computer & Electrical Engineering
and Computer Science
Florida Atlantic University
Boca Raton FL 33433

FAU
COLLEGE OF ENGINEERING & COMPUTER SCIENCE
Florida Atlantic University

CEN 5035 Software Engineering

Model Driven Development



Supporting literature

- Book: **Model-Driven Software Development: Technology, Engineering, Management (Paperback)**
  by Thomas Stahl, Markus Voelter, Krzysztof
  ISBN: 978-0-470-02570-3

- **Engineering Service Oriented Systems: A Model Driven Approach,** Karakostas, Bill; Zorgios, Yannis
  ISBN10: 1599049686 ISBN13: 9781599049687Cover: Hardcover April 2008

UML 2.0

- UML 2.0 and SysML Background and Reference material
- See www.uml-forum.com/specs.htm

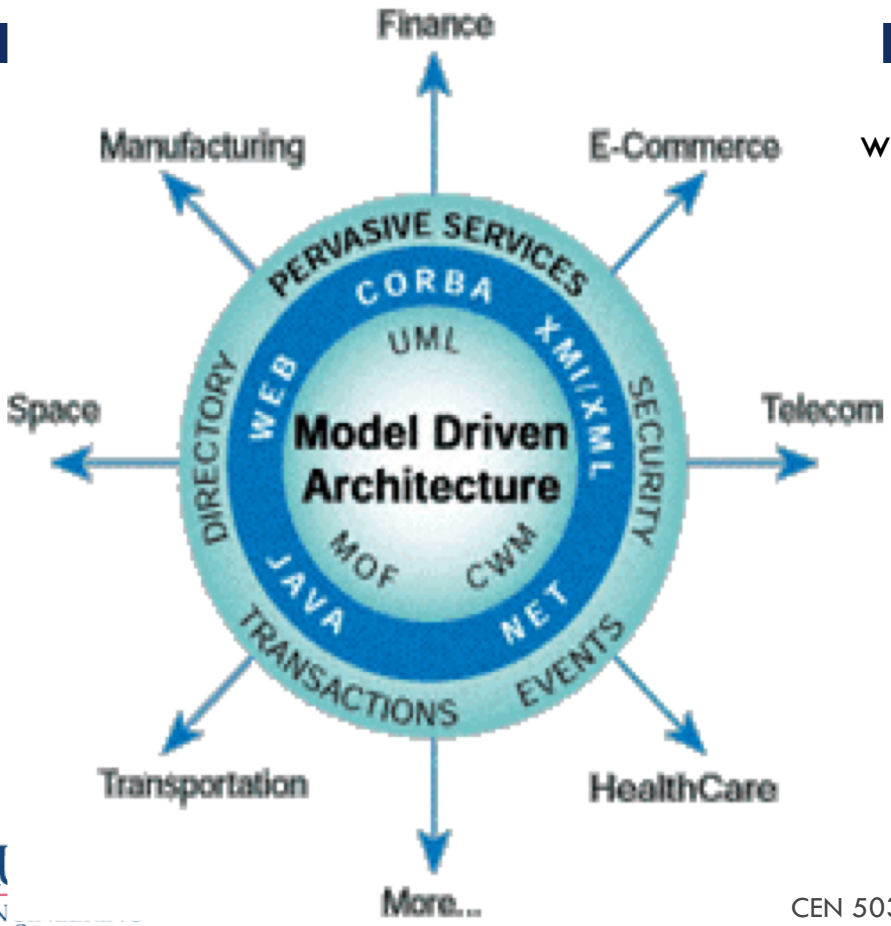
- OMG Standard
- http://www.omg.org/uml/ (UML)
- http://www.omg.org/mda/ (MDA)
- http://www.omg.org/cwm/ (MOF, XMI, CWM)

UML 2.0 Recommended Books

UML 2.0 in a Nutshell
by Dan Pilone (Author), Neil Pitman (Author)

The Unified Modeling Language User Guide Second 2nd Edition (ISBN 0-321-26797-4)
(G. Booch, J. Rumbaugh, I. Jacobsson)

OMG Model-Driven Architecture (MDA)



www.omg.org/mda

Model-driven – a definition

- A system development process is model-driven if
  - the development is mainly carried out using conceptual models at different levels of abstraction and using various viewpoints
  - it distinguishes clearly between platform independent and platform specific models
  - models play a fundamental role, not only in the initial development phase, but also in maintenance, reuse and further development
  - models document the relations between various models, thereby providing a precise foundation for refinement as well as transformation

MDA-Related Standards

- OMG Standards
  - Modeling – UML
  - Metamodeling – MOF
  - Action semantics
  - Model interchange – XMI
  - Diagram interchange
  - Human-readable textual notation – HUTN
  - Model-based testing and debugging
  - (CWM)
  - …
- Java Community Process (JCP) Standard
  - Java Metadata Interface – JMI

MDA technology standards

- **Unified Modeling Language (UML)**
  - UML is the de-facto standard industry language for specifying and designing software systems.
  - UML addresses the modelling of architecture and design aspects of software systems by providing language constructs for describing, software components, objects, data, interfaces, interactions, activities etc.
- **Meta Object Facility (MOF)**
  - MOF provides the standard modelling and interchange constructs that are used in MDA.
  - These constructs are a subset of the UML modelling constructs.
  - This common foundation provides the basis for model/metadata interchange and interoperability.
- **XML Metadata Interchange (XMI)**
  - XMI is a format to represent models in a structured text form.
  - In this way UML models and MOF metamodels may be interchanged between different modelling tools.
- **Common Warehouse Metamodel (CWM)**
  - CWM is the OMG data warehouse standard.
  - It covers the full life cycle of designing, building and managing data warehouse applications and supports management of the life cycle.
- **MOF Queries/View/Transformations (QVT)**
  - The goals of the QVT are to provide a standard specification of a language suitable for querying and transforming models which are represented according to a MOF metamodel.

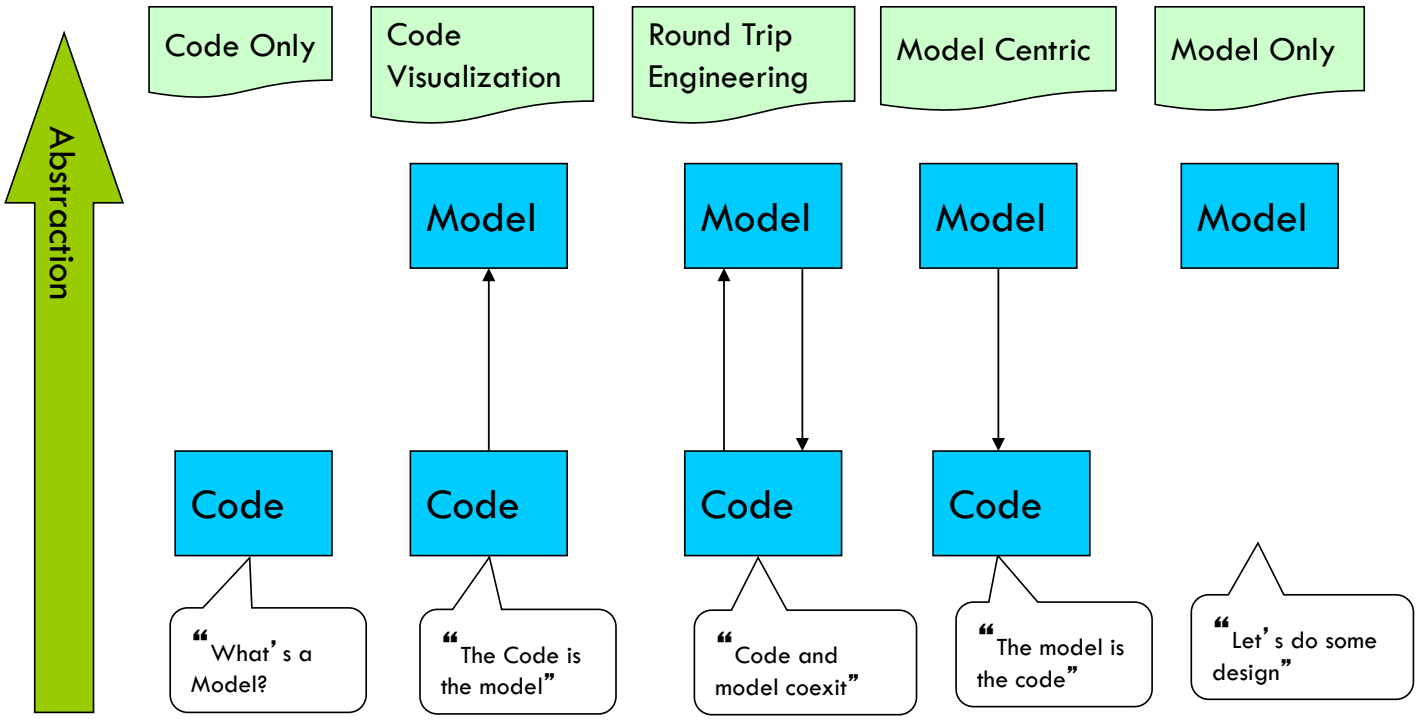
Basic concepts

- System
  - Existing or planned system.
  - System may include anything: a program, a single computer system, some combination of parts of different systems
- Model
  - A model of a system is a description or specification of that system and its environment for some certain purpose.
  - A model is often presented as a combination of drawings and text.
- Architecture
  - The architecture of a system is a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors.
  - MDA prescribes certain kinds of models to be used, how those models may be prepared and the relationships of the different kinds of models.
- Viewpoint
  - A viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system.
- View
  - A viewpoint model or view of a system is a representation of that system from the perspective of a chosen viewpoint.
- Platform
  - A platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independence

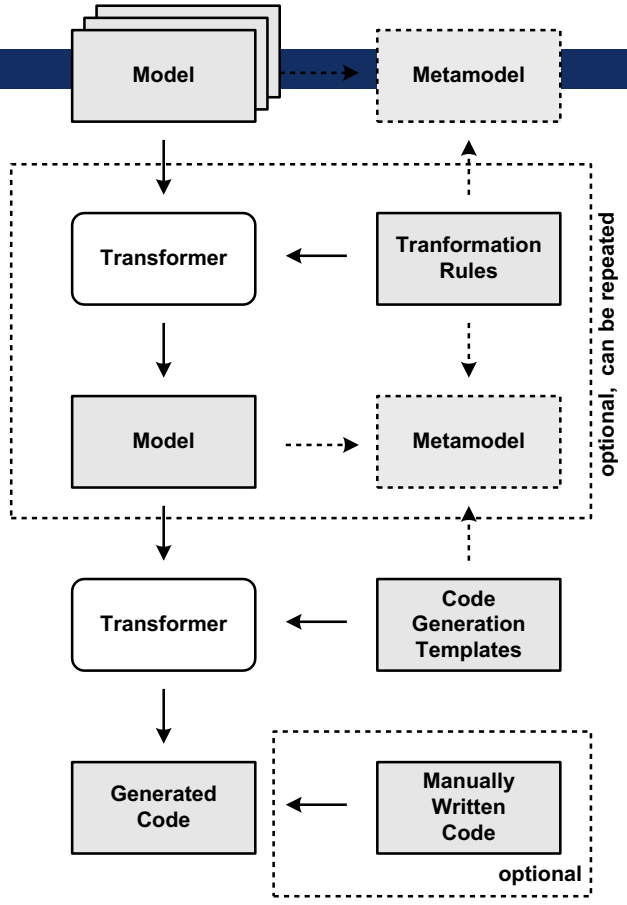
- The ability to develop platform independent models (PIM) is often quoted as a key advantage of MDE
- PIM capture the functionality of a system but don't depend on a specific implementation platform
- MDE support *multiple target platforms*
- The common functionality is modeling
- Transformations are used to generate implementations for the different platforms
- Reducing supporting multiple platforms and ensuring consistency across them

The Modeling Spectrum

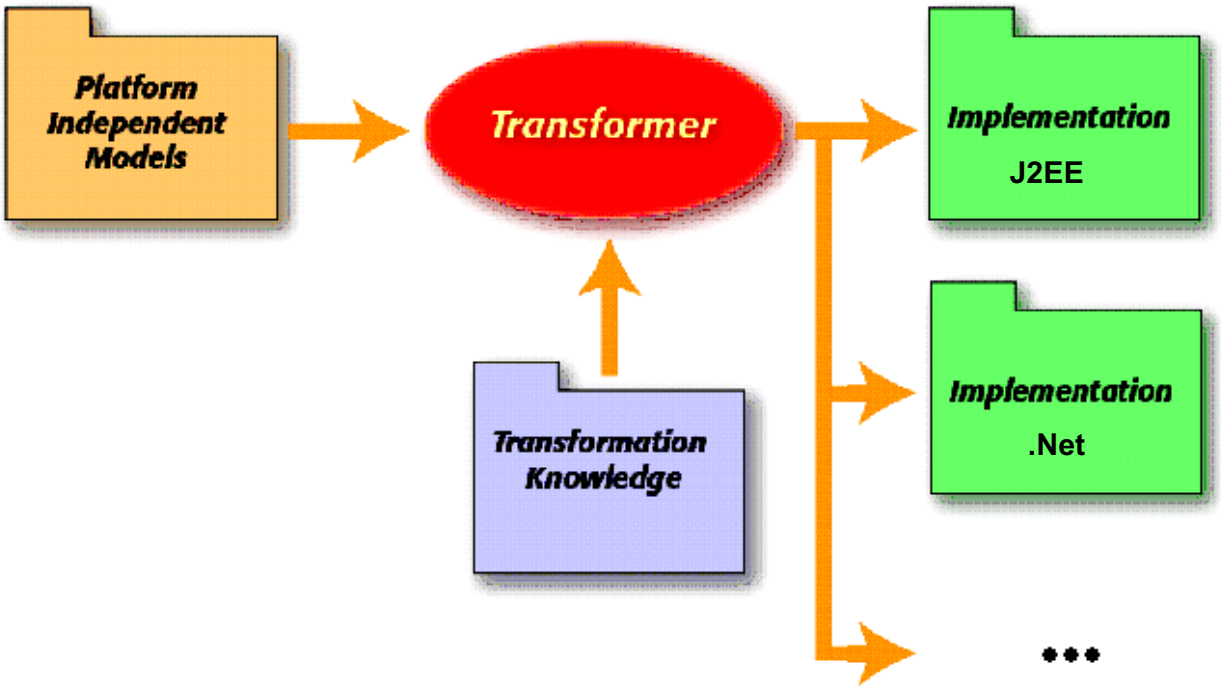


How does MDD work?

- Developer develops **model(s)** based on certain metamodel(s).
- Using **code generation templates,** the model is transformed to executable code.
- Optionally, the **generated code is merged** with manually written code.
- One or more **model-to-model transformation steps** may precede code generation.

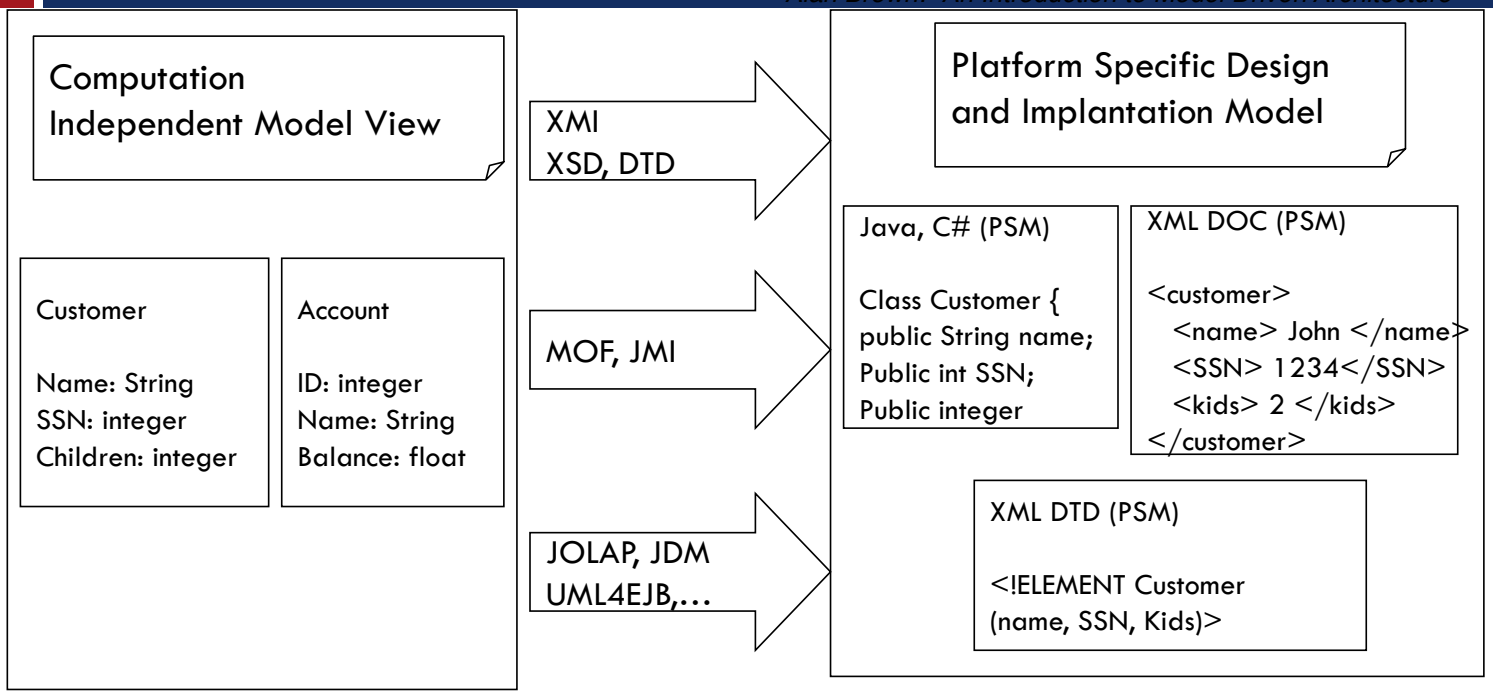


MDA: Transformation



- A PIM can be retargeted to different platforms
- Not the only reason why MDA might be of interest to you…

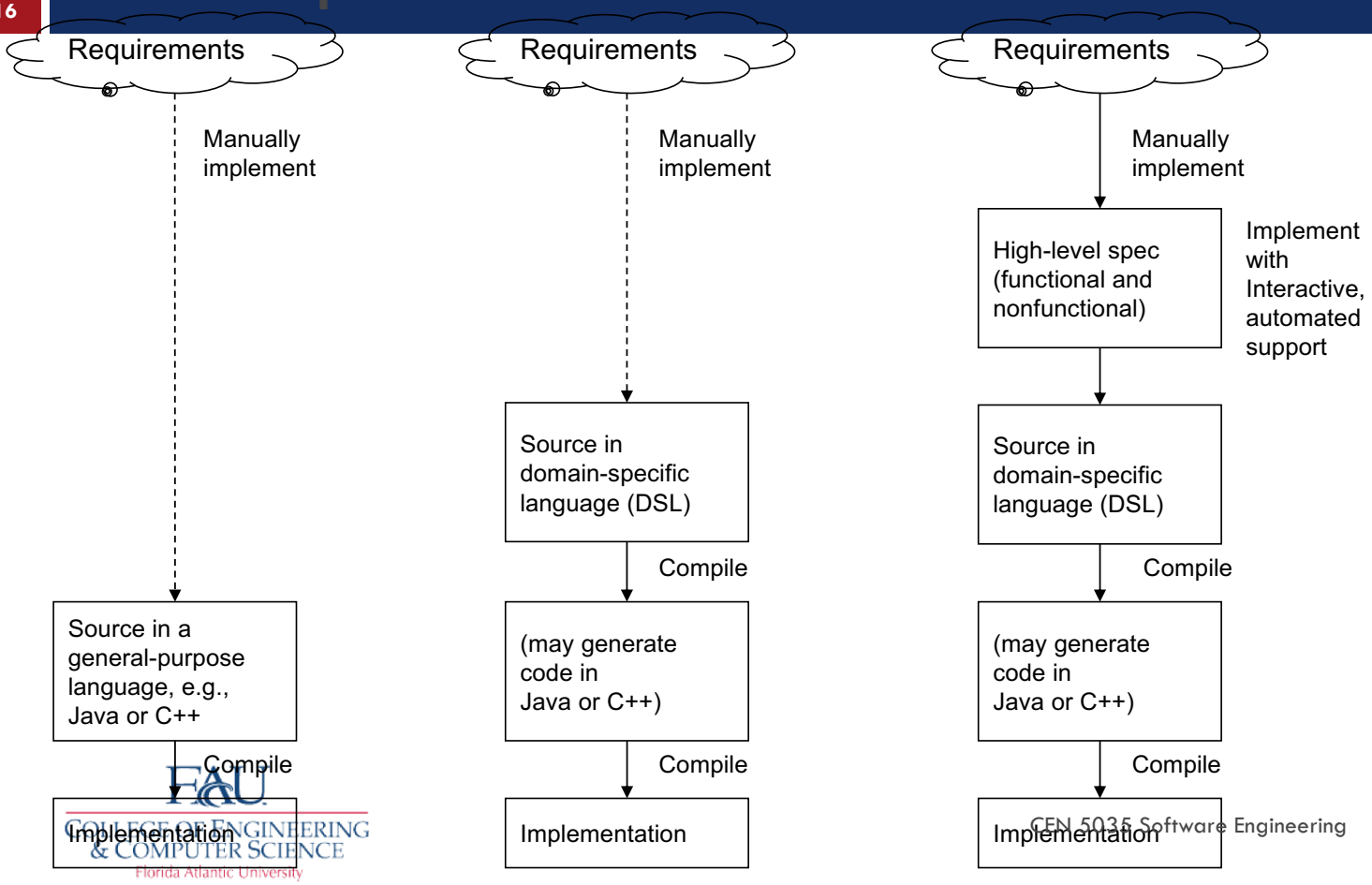
Transformation from PIM to PSM – A Simplified Example

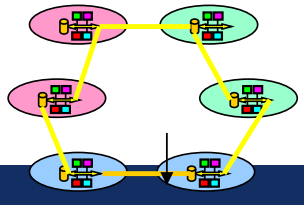


Platform Independent Models (PIM)

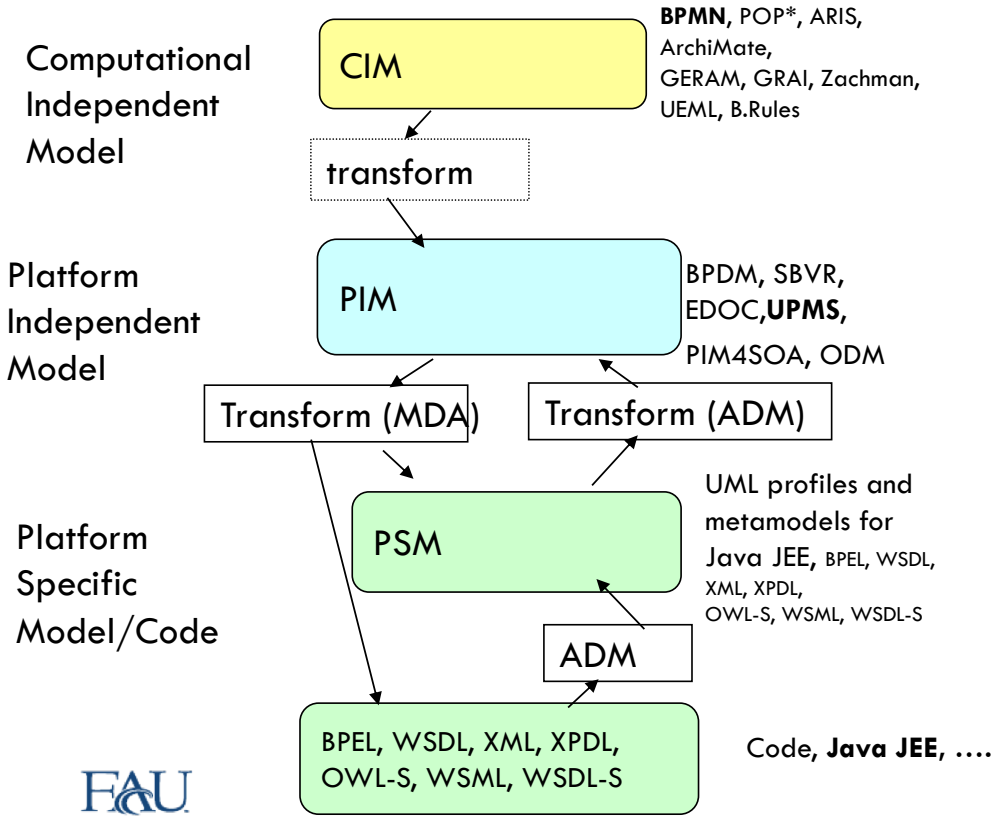
Platform Specific Models (PSM)

Automation in Software Development

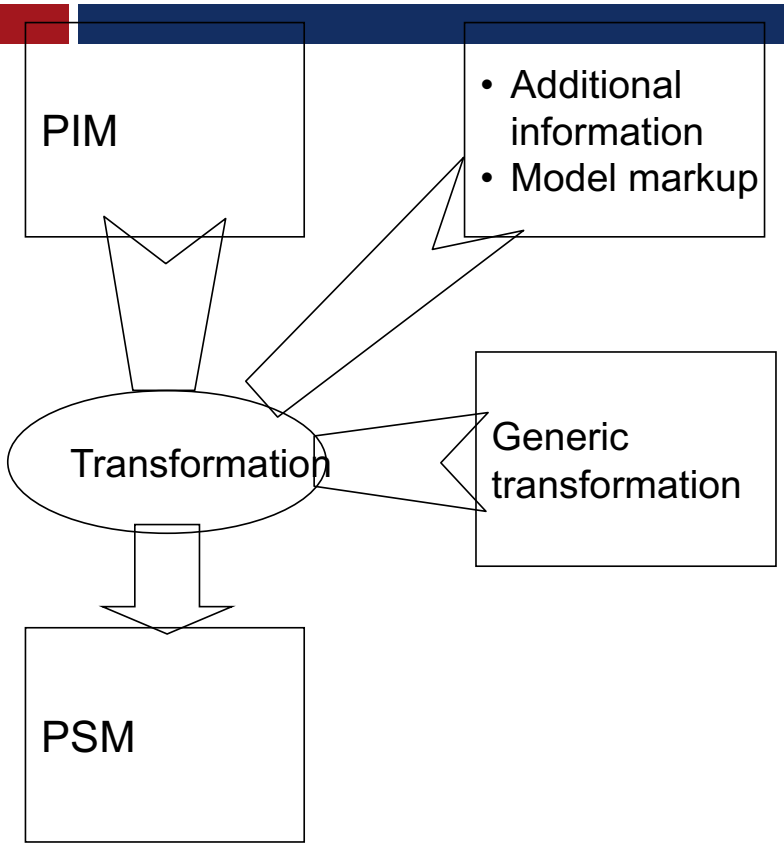




MDA
CIM, PIM and PSM/Code



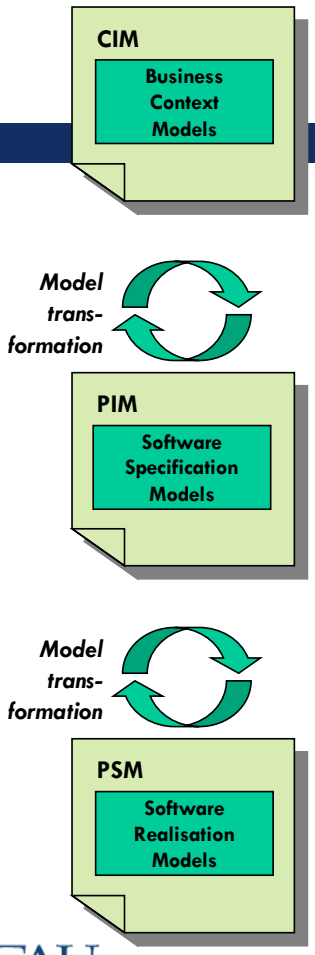
Basic MDA Pattern



- Generic transformations
  - Implement best practices, architectural and design patterns, technology patterns (e.g., J3EE patterns), optimizations, etc.
- Additional information
  - Adjust the transformation globally
  - Similar to compiler options
- Model markup
  - Direct the transformation of particular model elements
  - Not part of the PIM
  - Different platform mappings may require different markup
  - Similar to compiler programs

Goals

- The three primary goals of MDA are portability, interoperability and reusability.
- The MDA starts with the well-known and long established idea of separating the specification of the operation of the system from the details of the way the system uses the capabilities of its software execution platform (e.g. JEE, CORBA, Microsoft .NET and Web services).
- MDA provides an approach for:
  - specifying a system independently of the software execution platform that supports it;
  - specifying software execution platforms;
  - choosing a particular software execution platform for the system;
  - transforming the system specification into one for a particular software execution platform;



Model-driven approach to system engineering where models are used in

- understanding
- design
- construction
- deployment
- operation
- maintenance
- modification

Model transformation tools and services are used to align the different models.

Business-driven approach to system engineering where models are refined from business needs to software solutions

- Computation independent model (CIM) capturing business context and business requirements
- Platform independent model (PIM) focusing on software services independent of IT technology
- Platform specific model (PSM) focusing on the IT technology realisation of the software services

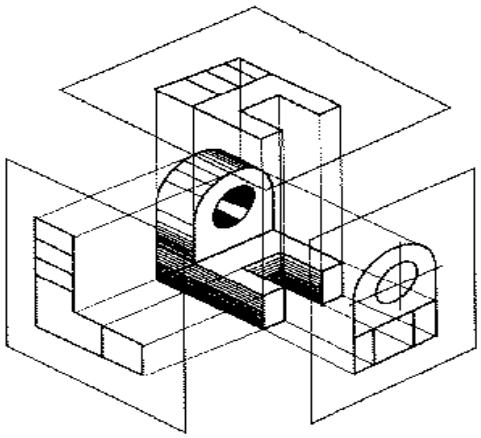
Matter of Relevance

- The basic pattern can be applied multiple times
- PIMs and PSMs are relative notions
  - "Someone's PIM can be someone else's PSM"
- Platform independence is relative, too
  - It's a scoping issue
  - It's a strategic decision

Role of Models

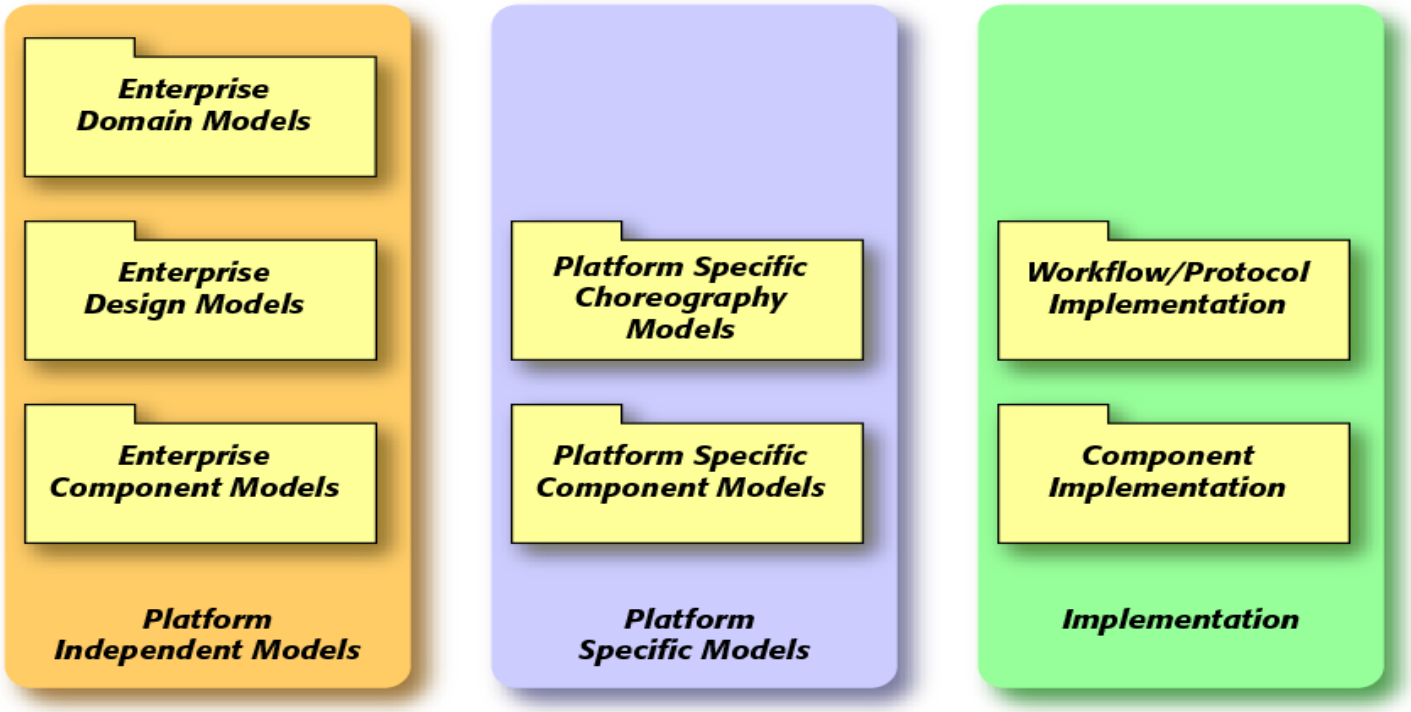
- Capture design information that is usually absent from code and lost during development
- Basis for
  - System generation
  - Analysis
  - Simulation
  - Test generation
  - Documentation generation
  - …
- *Domain-specificity* of a modeling language strengthens its capabilities for generation, optimization, early error detection, etc.

Viewpoints and Views



- System models are organized into multiple views
  - Different abstraction levels
  - Different aspects (e.g., workflow, domain concepts, deployment)
- Each view conforms to some viewpoint that prescribes some appropriate modeling notation
- Each viewpoint is relevant to some stakeholder

Many different views…



Benefits of MDA

- Preserving the investment in knowledge
  - Independent of implementation platform
  - Tacit knowledge made explicit
- Speed of development
  - Most of the implementation is generated
- Quality of implementation
  - Experts provide transformation templates
- Maintenance and documentation
  - Design and analysis models are not abandoned after writing
  - 100% traceability from specification to implementation

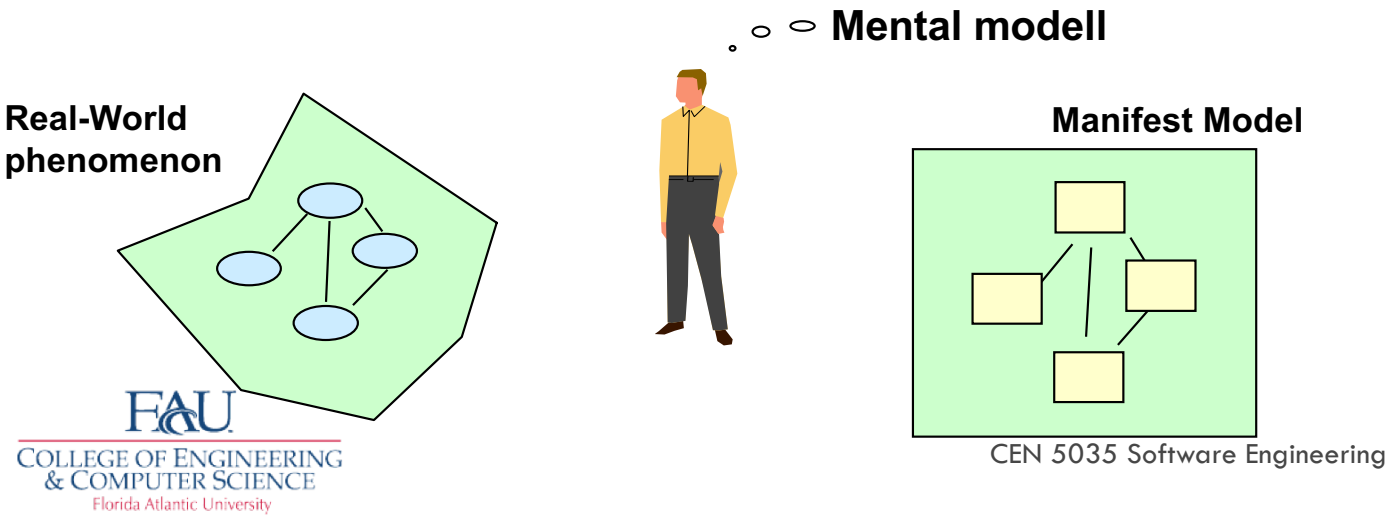
MDA – Three main abstraction levels

- **Computation independent model (CIM)**
  - The computational independent viewpoint is focused on the environment of the system and on the specific requirements of the system.
  - A CIM represents the computational independent viewpoint.
  - The CIM hides the structural details and, of course, the details related to the targeted platform.
- **Platform independent model (PIM)**
  - A platform independent model is a view of the system from a platform independent viewpoint.
  - The platform independent viewpoint is focused on the operation of the system, hiding the platform specific details.
  - A PIM exhibits platform independence and is suitable for use with a number of different platforms of similar types.
  - The PIM gathers all the information needed to describe the behaviour of the system in a platform independent way.
- **Platform specific model (PSM)**
  - A platform specific model is a view of the system from the platform specific viewpoint.
  - A PSM combines the specifications in the PIM with the details that specify how the system uses a particular type of platform.
  - The PSM represents the PIM taking into account the specific platform characteristics.

System and objects

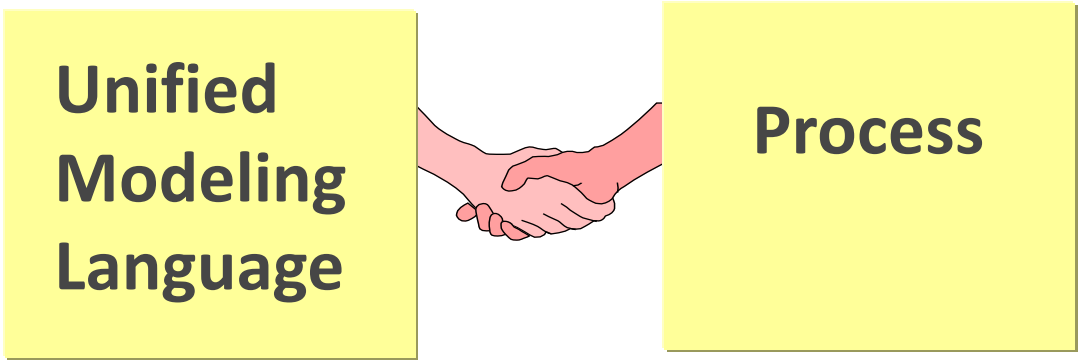
A *system* is a part of the real world which we choose to regard as a whole, separated from the rest of the world during some period of consideration.

A whole that we choose to consider as a collection of objects, each *object* being characterized by *attributes* and by *actions* which may involve itself and other objects.



UML and (R)UP

Two parts of a Harmonized Whole



- Convergence Today
- Unification leads to "standards"

- Convergence in the future
- Process frameworks through consensus

UML Structural Modeling

- Class Diagram
- Object Diagram
- Component Diagram (**new in UML 2.0**)
- Package Diagram
- Deployment diagram

UML Behavioral Modelling

- **Use Case Diagrams**
- Interactions
  - Sequence diagrams **(enhanced in UML 2.0)**
  - Timing diagrams **(new in UML 2.0)**
  - Interaction overview diagrams **(new in UML 2.0)**
  - Communication diagrams (i.e. collaboration diagram)
- State machine diagrams **(enhanced in UML 2.0)**
- Activity Diagrams **(enhanced in UML 2.0)**

Different kind of models

- Conceptual models
- Specification models
- Implementation models

UML Information Modeling

- Ref also ISO 19103 Standard for Conceptual Modeling
- The following material is for reference …..

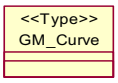
ISO 19103 – Conceptual modeling with UML

Implementation neutral! Not implementation specific!

- Basic data types
- Stereotypes
- Naming
- Documentation of models
- Information modelling guidelines

Documentation of models

- Diagrams
  - Package dependency diagrams
  - Class diagrams
  - Class context diagrams
- Text
  - Class
    - General description, semantics, supertypes/subtypes
  - Attributes, Associations
  - Operations
    - preconditions, input/output parameters, return value, post conditions, exceptions, constraints
  - Constraints



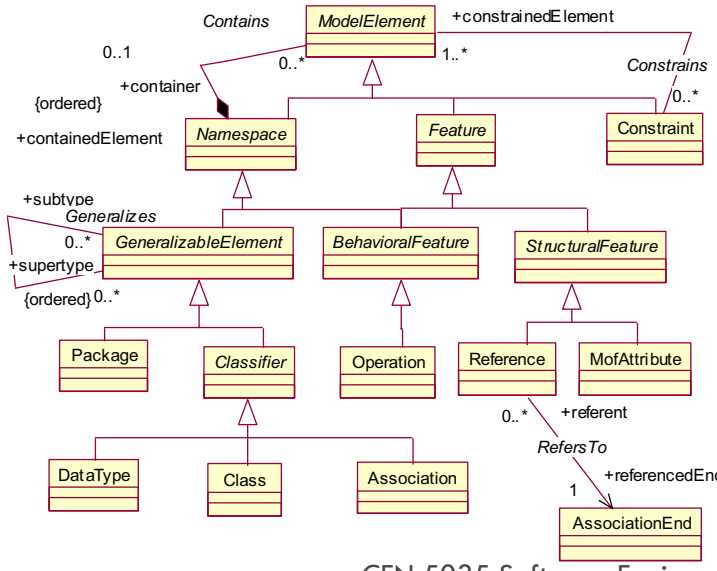
NB! Font size > 8pt

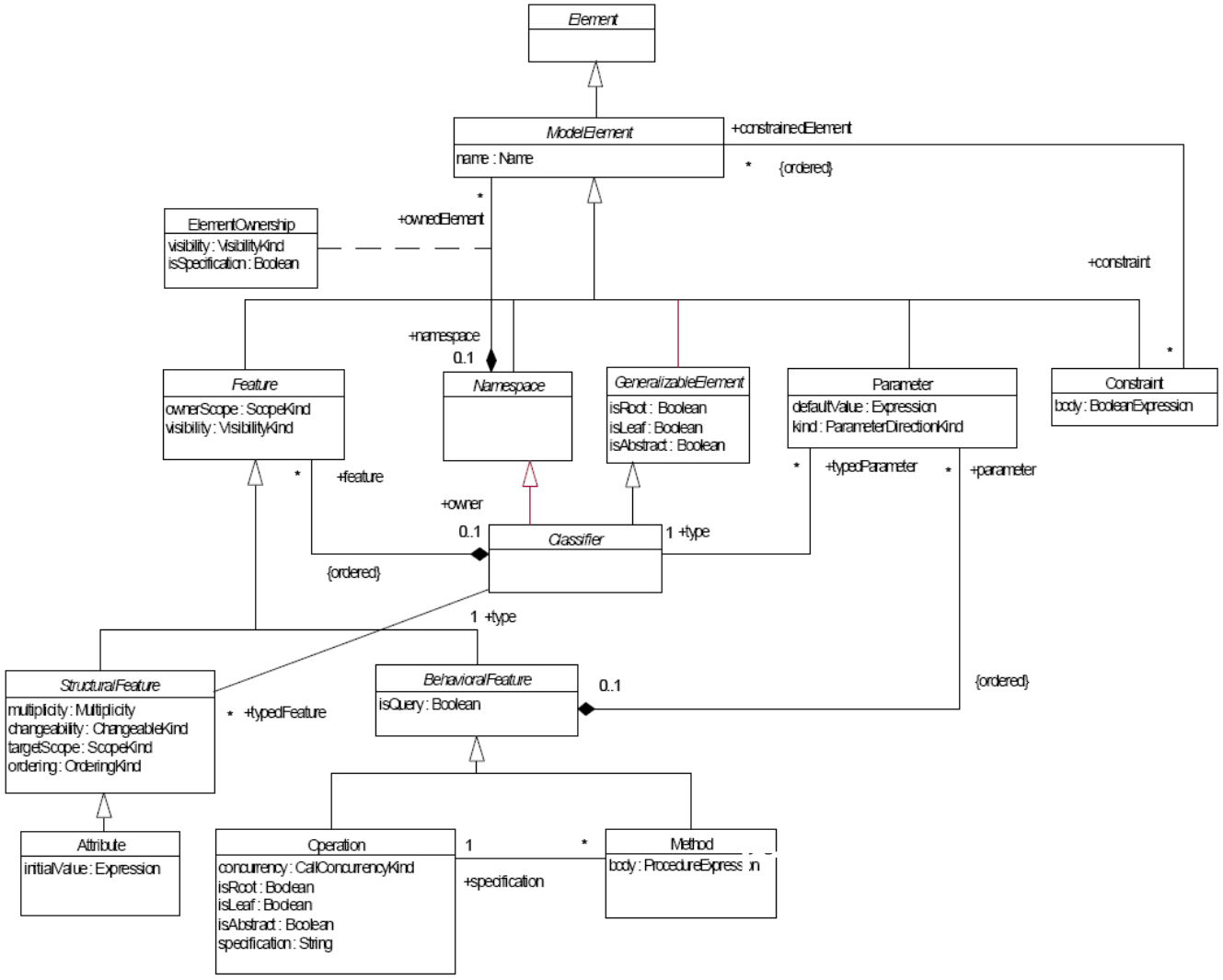
Domain specific modelling languages

- Specific to a domain
- More focussed purpose
- Usable by the domain experts
- More productive than general purpose
  - If properly designed and tooled!
- UML profiles vs. DSL

Assigning Meaning to Models

- If a model *is no longer* just
  - fancy pictures to decorate your room
  - a graphical syntax for C++/Java/C#/Eiffel...
- Then tools must be al
  - Let's make a model of what a model is!
  - => ***meta-modeling***
    - & meta-meta-modeli
    - Use Meta-Object Facility (MOF) to avo infinite Meta-recursio




Element
ModelElement
name : Name
+constrainedElement
{ordered}
ElementOwnership
visibility : VisibilityKind
isSpecification : Boolean
+ownedElement
+constraint
+namespace
0..1
Feature
ownerScope : ScopeKind
visibility : VisibilityKind
Namespace
GeneralizableElement
isRoot : Boolean
isLeaf : Boolean
isAbstract : Boolean
Parameter
defaultValue : Expression
kind : ParameterDirectionKind
Constraint
body : BooleanExpression
+feature
+owner
0..1
Classifier
1 +type
{ordered}
+typedParameter
+parameter
1 +type
StructuralFeature
multiplicity : Multiplicity
changeability : ChangeableKind
targetScope : ScopeKind
ordering : OrderingKind
+typedFeature
BehavioralFeature
isQuery : Boolean
0..1
{ordered}
Attribute
initialValue : Expression
Operation
concurrency : CallConcurrencyKind
isRoot : Boolean
isLeaf : Boolean
isAbstract : Boolean
specification : String
1
+specification
Method
body : ProcedureExpression

Generalizations

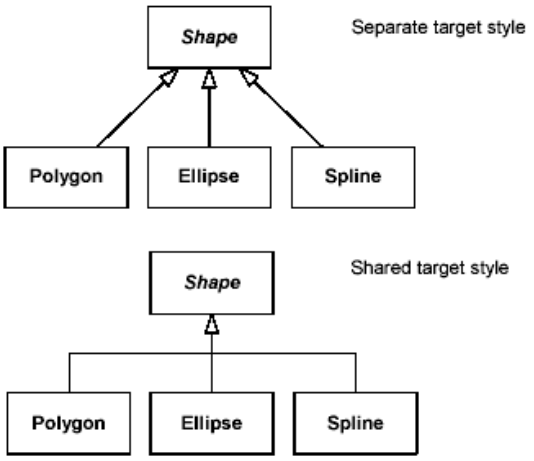


Figure 3-33. Examples of generalizations between classes.

NB: Tell you nothing about:
- *generalization being acyclic,*
- *or semantics of dynamic binding*

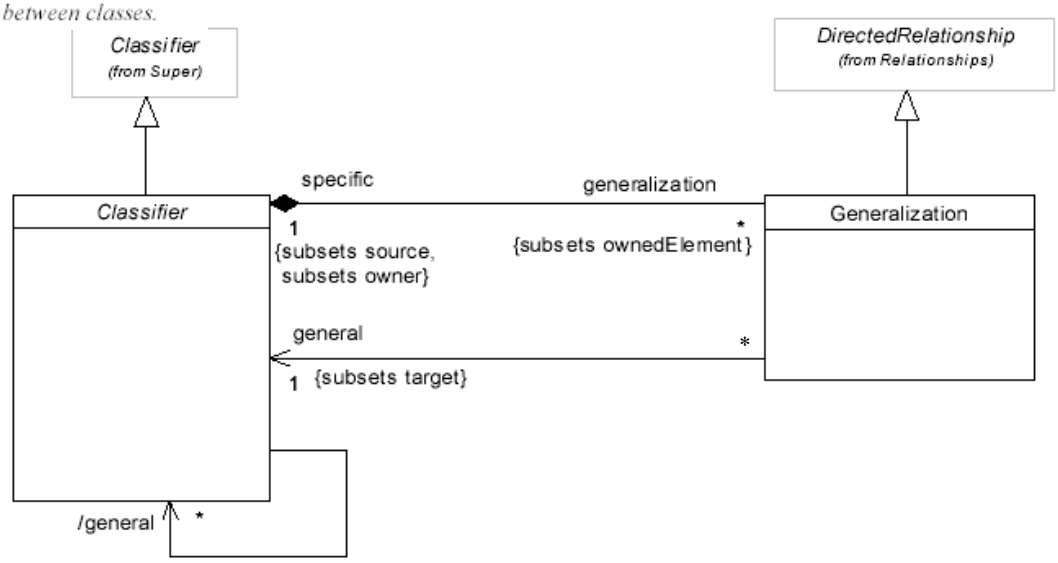


Figure 3-32. The elements defined in the Generalizations package.

Goals & Challenges

- **Goals:**
  - We need an **end-to-end tool chain** that allows us to build models, verify them and generate various artefacts from them.
  - All of this should happen in a homogeneous environment, namely Eclipse.
- **Challenges:**
  - **Good Editors** for your models
  - **Verifying** the models as you build them
  - **Transforming/Modifying** models
  - **Generating** Code
  - **Integrating** generated and non-generated code

Some historic references

[Atkinson and Kühne 2003] C. Atkinson and T. Kühne, "Model-Driven Development: A Metamodeling Foundation", IEEE Software, vol. 20, no. 5, pp. 36-41, 2003. http://www.mm.informatik.tu-darmstadt.de/staff/kuehne/publications/papers/mda-foundation.pdf

[Clark, et al. 2004] T. Clark, A. Evans, P. Sammut, and J. Willans, "Applied Metamodelling - A Foundation for Language Driven Development, Version 0.1", 2004. http://albini.xactium.com/web/index.php?option=com_remository&Itemid=54&func=select&id=1

[Seidewitz 2003] E. Seidewitz, "What Models Mean", IEEE Software, vol. 20, no. 5, pp. 26-32, 2003.

[Swithinbank, et al. 2005] P. Swithinbank, M. Chessell, T. Gardner, C. Griffin, J. Man, H. Wylie, and L. Yusuf, "Patterns: Model-Driven Development Using IBM Rational Software Architect", IBM, Redbooks, December 2005. http://www.redbooks.ibm.com/redbooks/pdfs/sg247105.pdf

Contact

Dr. Shihong Huang
shihong@fau.edu
Dept. of Computer & Electrical Engineering and Computer Science
Engineering East (EE96) Room 434
Florida Atlantic University
Boca Raton FL 33433

FAU
COLLEGE OF ENGINEERING & COMPUTER SCIENCE
Florida Atlantic University