MDA-based Development in Practice

Uniting Model and Code So That They Can‘t Drift Apart

Ruud Grotens, Product Manager, Compuware

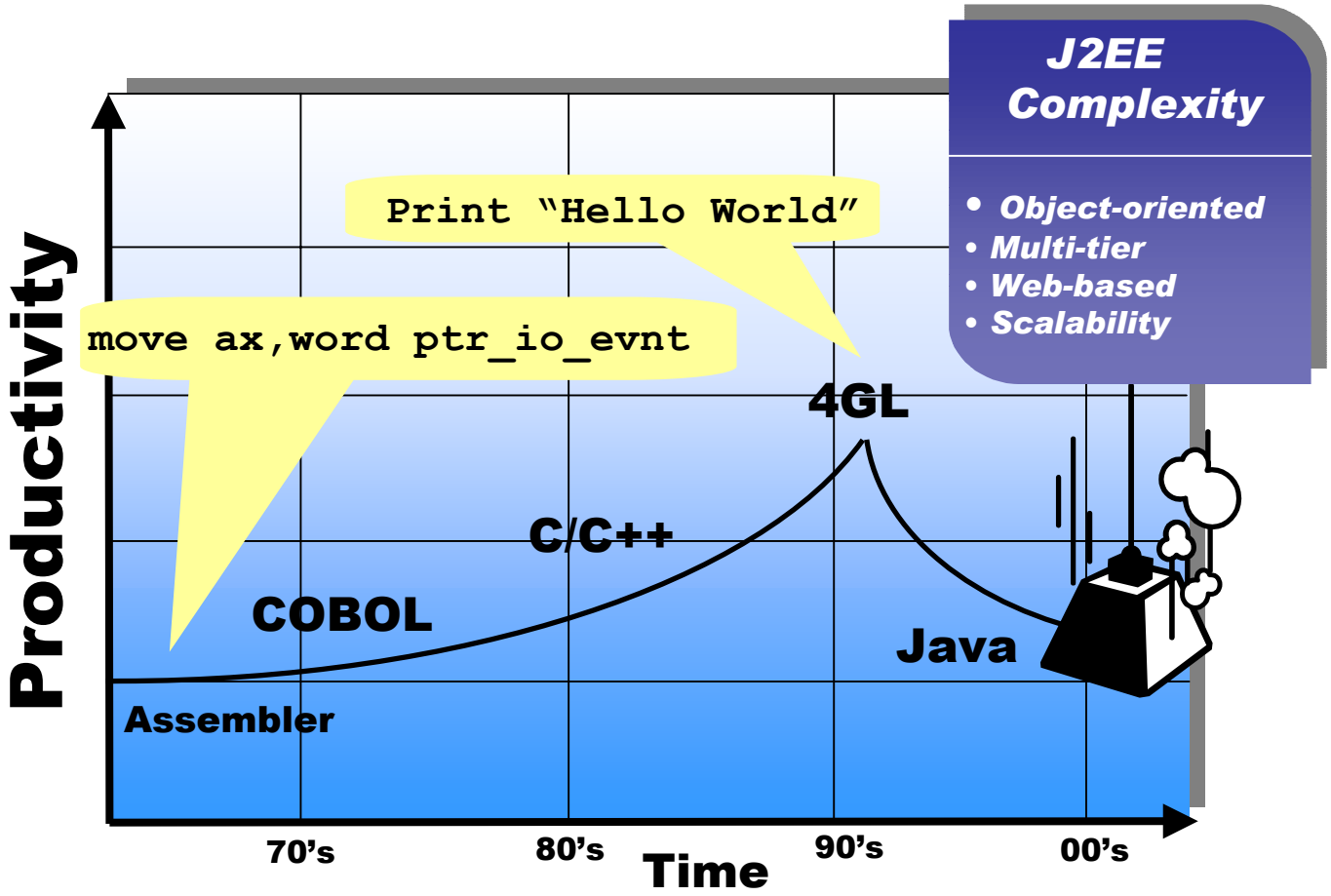
COMPUWARE

"The entire history of software engineering is that of the rise in levels of abstraction."

Grady Booch

The limits of Software, September 2002

Coding Language Evolution



Raise the level of Abstraction

Modeling

- Many practice modeling in some form
  - ‘Bubble and arrow’ diagrams
  - Flow charts, dataflow diagrams, state transition diagrams
- UML: The Standard modeling language

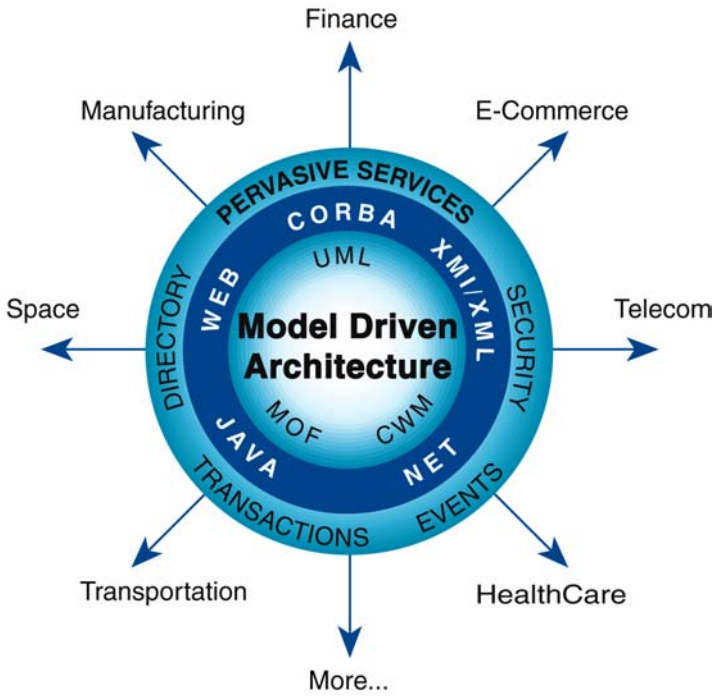
But …

- Models are not necessarily on a higher level of abstraction than code
  - Diagrammatical presentation of the code structure
    - e.g. UML class diagram of Java classes
    - No increased productivity because of equal complexity

- Modeling is only *half* the answer
  - The UML model is tied to a hardware/operating architecture
  - UML model is implementation-specific

Model Driven Architecture

Raising the Level of Abstraction

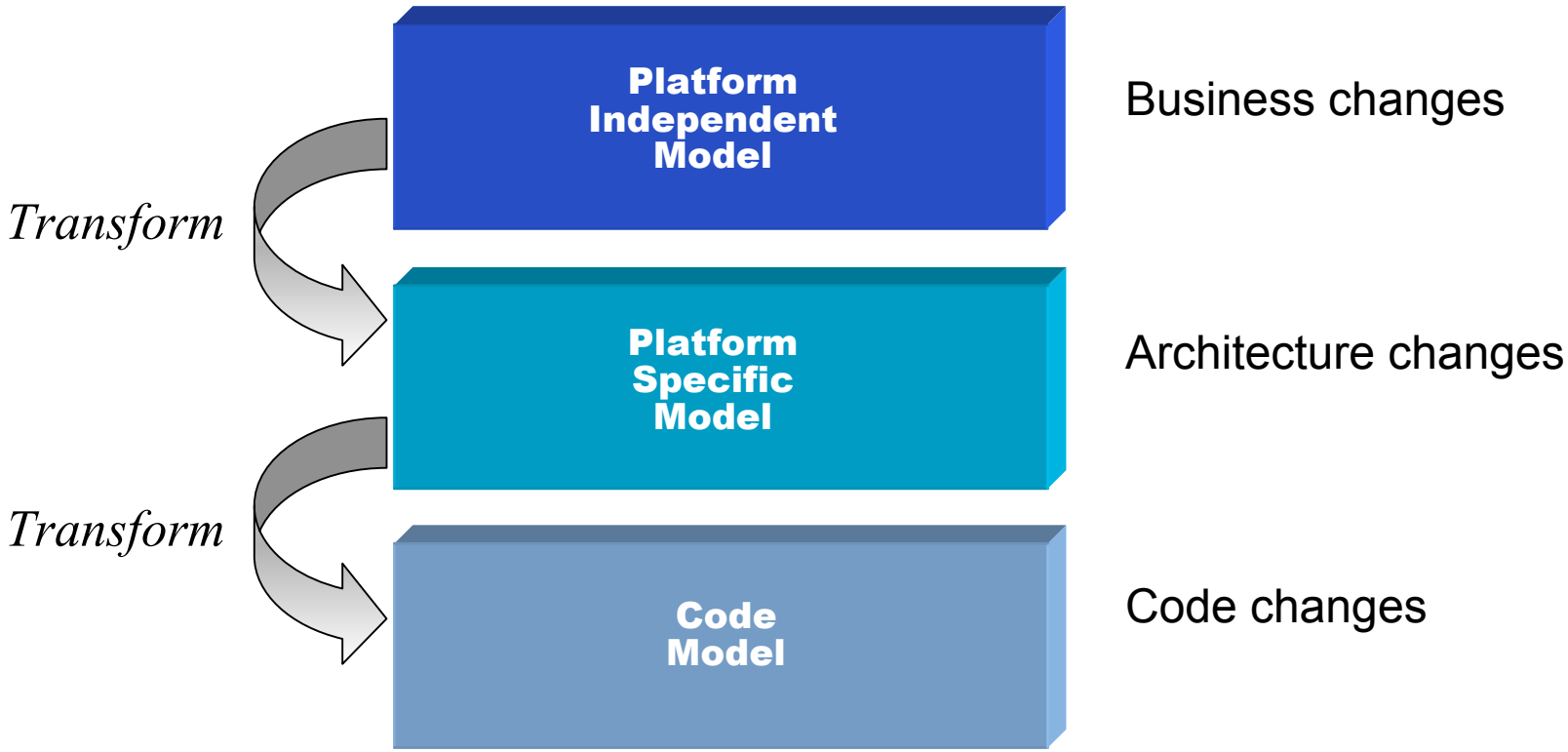


Object Management Group (OMG)
www.omg.org

- Invest in models as the long-lived intellectual assets, not code
  - Raise abstraction level above deployment platform
- Model Driven Architecture
  - Modelling instead of programming
  - Merging modelling and coding
  - Based on standards
    - UML, MOF, CWM, XMI ...

MDA Development

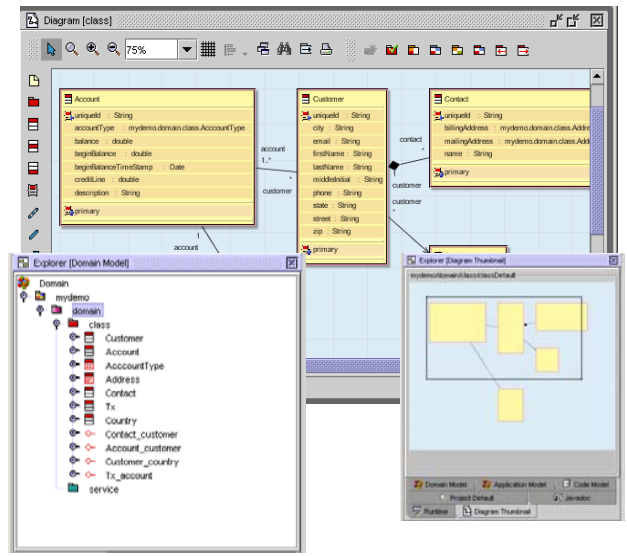
Generation of working applications



Platform Independent Model

Step One

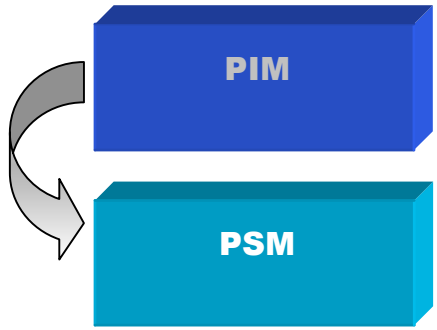
- High level of abstraction
  - Business centric
  - Independent of any implementation technology or technology details
- Expressed in UML
  - Business functionality
  - Structural aspects
- Enrich with business rules
  - Constraints
  - Expressions
- Automated reuse of model definitions and business rules in lower level models

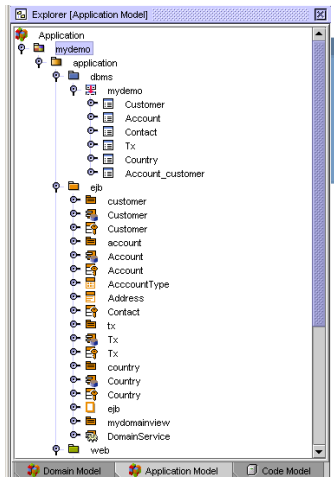


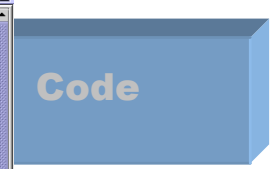
Platform Specific Model

Step Two

- Automatic transformation from *PIM* to *PSM*
  - One or more target platforms
  - PIM and PSM always in sync
- Focus on specific implementation technology
  - Presentation Model (Web)
    - Data schemas, web components, etc.
  - Business Logic Model (EJB)
    - Data schemas, key classes, entity components, session components, etc.
  - Data Model (DBMS)
    - Relational data schema, Tables, columns, keys, etc.
- Hiding complexity in abstract specification
  - Incremental Refinement



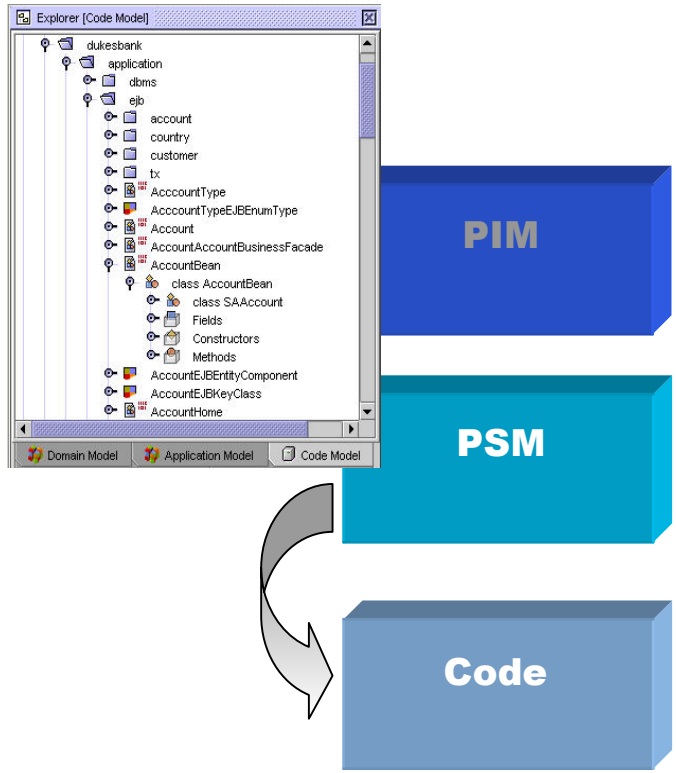




Code Model

Step Three

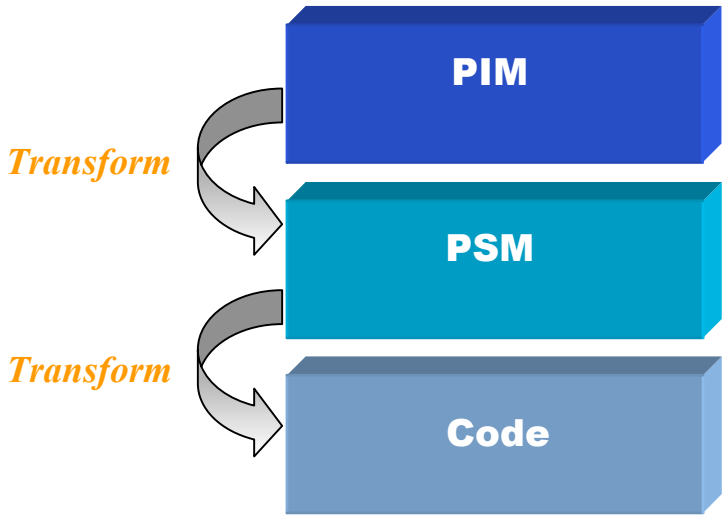
- Automatic transformation from *PSM* to *Code*
  - PSM and Code always in sync
- Complete executable results
  - Presentation Tier
    - JSP, Servlets
  - EJB Tier
    - Bean class, home/remote and primary key classes etc.
  - DBMS Tier
    - SQL scripts for target database
  - Application deployment descriptors
    - For target application server
- Code customization in ‘Free Blocks’



Transformation of Models

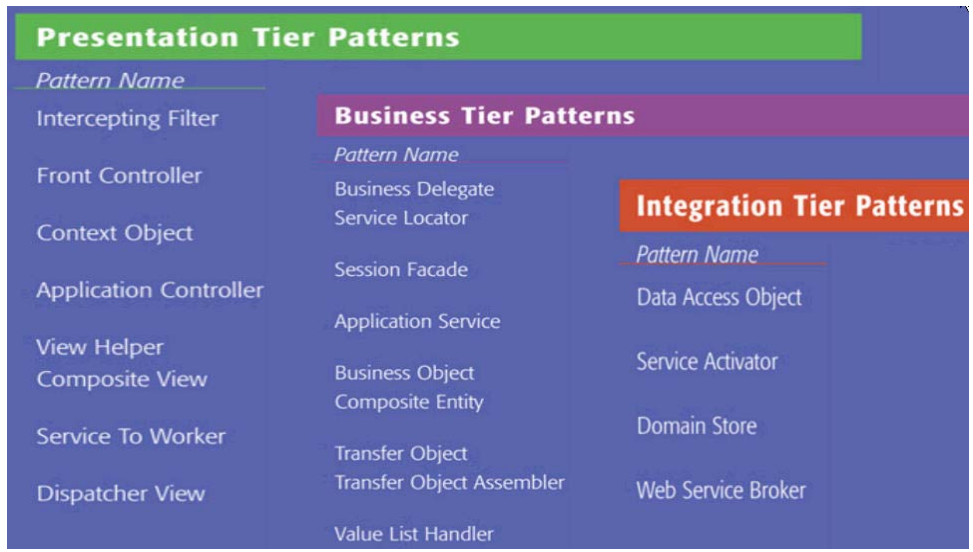
Bridging the Quality Gap

- Delivering solutions through a combination of formal *Models* and solution *Patterns*
- Models and Patterns are complementary
  - Models provide abstraction
    - Focus on building future proof applications
    - Reducing business complexity
  - Patterns provide best practices
    - Hides the implementation through abstraction
    - Reducing technology complexity



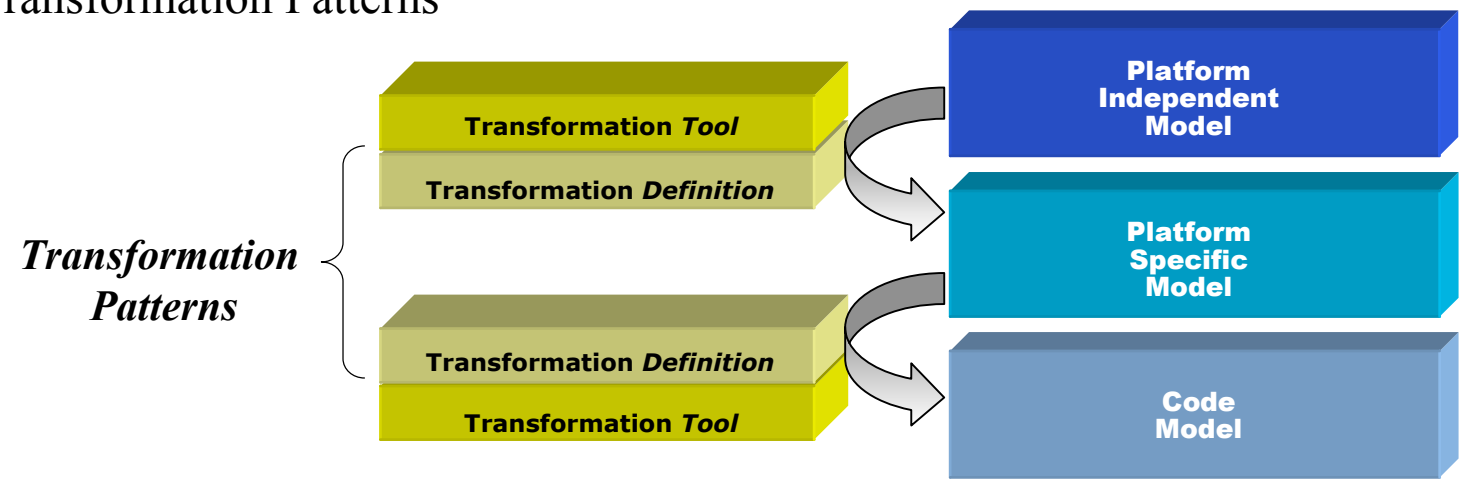
The Emergence of Patterns

- Recurring solutions to similar problems
  - Patterns come from good designs
- Patterns can cover a wide range of abstractions
  - Code-level
  - Architecture-level
  - Business-level
- The “Gang of Four”
  - Non-platform patterns
- SUN J2EE Design Patterns
  - Multi level patterns



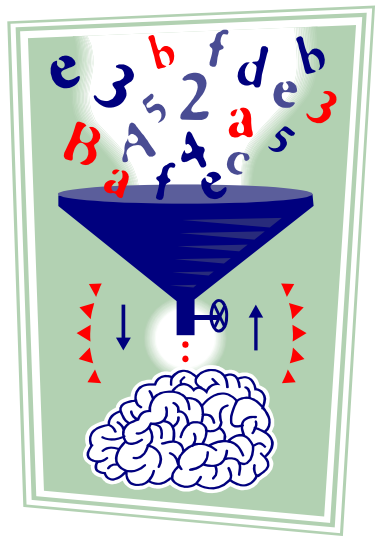
How to use Models and Patterns to build an application?

- Transformations are essential in the MDA development process
  - Transformations *between models* and *between models and code*
- Transformations are always executed by tools
  - Transformation Definitions
- MDA in Practice
  - Standards based
  - Repository based (MOF)
  - Transformation Patterns



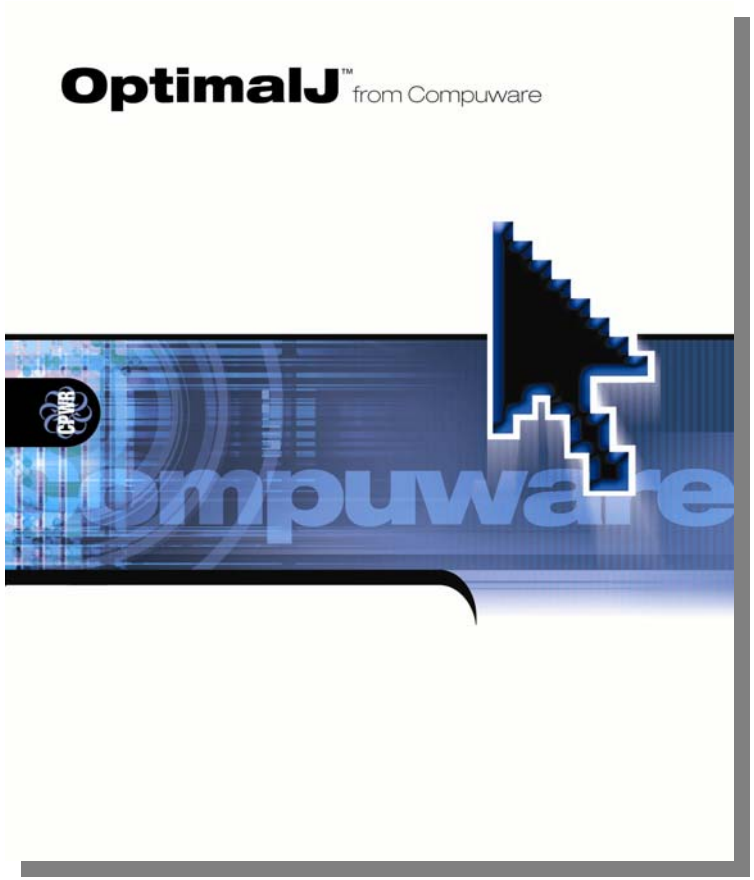
Productivity & Control

- The Reality is …
  - MDA is *not* just another CASE tool
    - MDA is based on standards only!
    - Non proprietary solution!
  - MDA is *not* just another Black-box code generator
    - Customizable and Maintainable Transformations!
    - Full control over generated code!
- Enterprises usually have clear ideas on how to implement a given model, based on:
  - Architectural and Coding Standards, Best Practices
  - Past experience - Patterns come from good designs
- Enforcement of standards and best practices
  - e.g. Automated implementation of J2EE Design Patterns



MDA in Practice

Demonstration

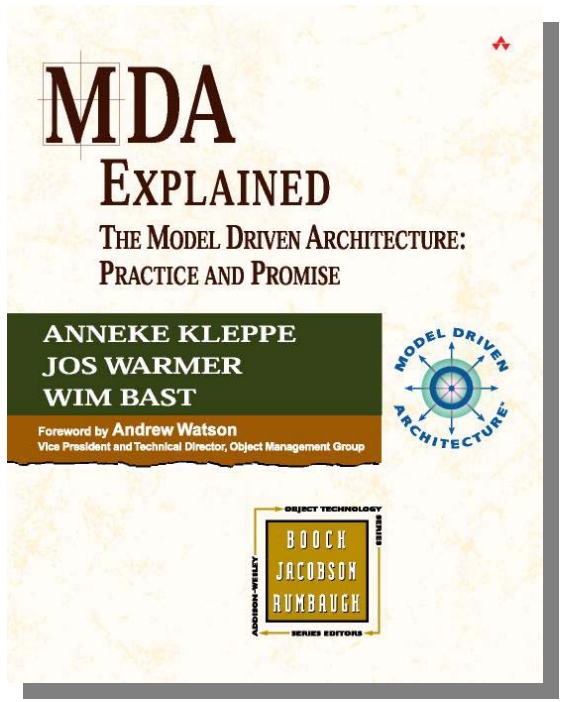


Raising the Level of Abstraction is MDA

- Platform Independent Models as the long-lived intellectual assets, not code
- Raise abstraction level above deployment platform
- Increased productivity because of automated transformations
- Quality improvement because of enforcement of standards and best practices
- Always in control of the generated application

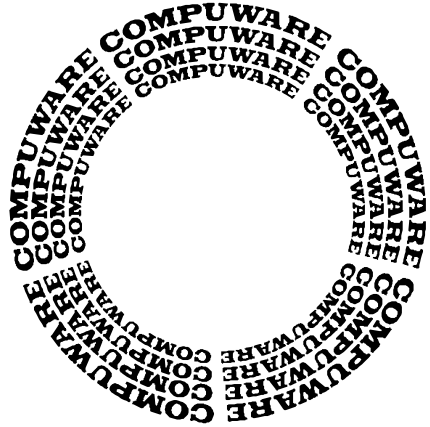
Find Out More …

- OMG's MDA web site: www.omg.org/mda
  - Compuware's MOF 2.0 Query/View/ Transformations (QVT) RfP
    - www.omg.org/cgi-bin/doc?ad/03-03-24
    - www.omg.org/cgi-bin/doc?ad/02-09-12
- OptimalJ web site
  - www.compuware.com/optimalj
  - http://javacentral.compuware.com
    - Technical Forums
    - Technical white papers
- ***MDA Explained***
  - Addison-Wesley ISBN: 0-321-19442-X



Contact Details

- Compuware Nordic AS
  - Pål Christoffersen
  - Email: *Paal.Christoffersen@compuware.com*
- Compuware Europe B.V.
  - *Ruud.Grotens@compuware.com*



COMPUWARE®